Copyright © Huawei Technologies Co., Ltd. 2020. All rights reserved.
No part of this document may be reproduced or transmitted in any form or by any means without prior
written consent of Huawei Technologies Co., Ltd.

Trademarks and Permissions

and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and
the customer. All or part of the products, services and features described in this document may not be
within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements,
information, and recommendations in this document are provided "AS IS" without warranties, guarantees
or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the
preparation of this document to ensure accuracy of the contents, but all statements, information, and
recommendations in this document do not constitute a warranty of any kind, express or implied.
Contents

1 Overview.................................................................1
2 Downloading Related Tools........................................3
3 gsql..............................................................................6
  3.1 gsql Overview.......................................................6
  3.2 Instruction.............................................................14
  3.3 Online Help...........................................................16
  3.4 Command Reference..............................................17
  3.5 Meta-Command Reference......................................22
  3.6 Troubleshooting.....................................................41
4 Data Studio - Integrated Database Development Tool........46
  4.1 About Data Studio..................................................46
  4.1.1 Overview............................................................46
  4.1.2 Constraints and Limitations.................................47
  4.1.3 Structure of Release Package...............................49
  4.1.4 System Requirements.........................................51
  4.2 Installing and Configuring Data Studio.....................52
  4.3 Getting Started.......................................................61
  4.4 Data Studio User Interface......................................62
  4.5 Data Studio Menus..................................................63
  4.5.1 File........................................................................63
  4.5.2 Edit........................................................................64
  4.5.3 Run.........................................................................68
  4.5.4 Debug.................................................................69
  4.5.5 Settings...............................................................69
  4.5.6 Help.......................................................................70
  4.6 Data Studio Toolbars..............................................71
  4.7 Data Studio Right-Click Menus...............................71
  4.8 Connection Profiles.................................................77
  4.8.1 Overview............................................................77
  4.8.2 Adding a Connection..........................................77
  4.8.3 Renaming a Connection......................................85
  4.8.4 Editing a Connection..........................................85

Copyright © Huawei Technologies Co., Ltd.
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.20.2 Working with Users/Roles</td>
<td>176</td>
</tr>
<tr>
<td>4.21 SQL Terminal</td>
<td>177</td>
</tr>
<tr>
<td>4.21.1 Opening SQL Terminals</td>
<td>177</td>
</tr>
<tr>
<td>4.21.2 Managing SQL Query Execution History</td>
<td>182</td>
</tr>
<tr>
<td>4.21.3 Opening And Saving SQL Scripts</td>
<td>184</td>
</tr>
<tr>
<td>4.21.4 Viewing Object Properties in the SQL Terminal</td>
<td>186</td>
</tr>
<tr>
<td>4.21.5 Canceling Execution of SQL Queries</td>
<td>188</td>
</tr>
<tr>
<td>4.21.6 Formatting of SQL Queries</td>
<td>188</td>
</tr>
<tr>
<td>4.21.7 Selecting a DB Object in the SQL Terminal</td>
<td>194</td>
</tr>
<tr>
<td>4.21.8 Viewing the Query Execution Plan and Cost</td>
<td>195</td>
</tr>
<tr>
<td>4.21.9 Viewing the Query Execution Plan and Cost Graphically</td>
<td>198</td>
</tr>
<tr>
<td>4.21.10 Working with the SQL Terminals</td>
<td>201</td>
</tr>
<tr>
<td>4.21.11 Exporting Query Results</td>
<td>217</td>
</tr>
<tr>
<td>4.21.12 Managing SQL Terminal Connections</td>
<td>218</td>
</tr>
<tr>
<td>4.22 Batch Operation</td>
<td>219</td>
</tr>
<tr>
<td>4.22.1 Overview</td>
<td>219</td>
</tr>
<tr>
<td>4.22.2 Dropping Batch of Objects</td>
<td>220</td>
</tr>
<tr>
<td>4.22.3 Granting/Revoking Privileges</td>
<td>221</td>
</tr>
<tr>
<td>4.23 Personalizing Data Studio</td>
<td>222</td>
</tr>
<tr>
<td>4.23.1 Overview</td>
<td>222</td>
</tr>
<tr>
<td>4.23.2 General</td>
<td>222</td>
</tr>
<tr>
<td>4.23.3 Editor</td>
<td>226</td>
</tr>
<tr>
<td>4.23.4 Environment</td>
<td>231</td>
</tr>
<tr>
<td>4.23.5 Export</td>
<td>233</td>
</tr>
<tr>
<td>4.23.6 Result Management</td>
<td>234</td>
</tr>
<tr>
<td>4.23.7 Security</td>
<td>239</td>
</tr>
<tr>
<td>4.24 Performance Specification</td>
<td>242</td>
</tr>
<tr>
<td>4.25 Security Management</td>
<td>243</td>
</tr>
<tr>
<td>4.25.1 Overview</td>
<td>244</td>
</tr>
<tr>
<td>4.25.2 Login History</td>
<td>244</td>
</tr>
<tr>
<td>4.25.3 Password Expiry Notification</td>
<td>244</td>
</tr>
<tr>
<td>4.25.4 Securing the Application In-Memory Data</td>
<td>244</td>
</tr>
<tr>
<td>4.25.5 Data Encryption for Saved Data</td>
<td>245</td>
</tr>
<tr>
<td>4.25.6 SQL History</td>
<td>245</td>
</tr>
<tr>
<td>4.25.7 SSL Certificates</td>
<td>245</td>
</tr>
<tr>
<td>4.26 Troubleshooting</td>
<td>252</td>
</tr>
<tr>
<td>4.27 FAQs</td>
<td>258</td>
</tr>
<tr>
<td>5 GDS - Parallel Data Loader</td>
<td>265</td>
</tr>
<tr>
<td>5.1 Installing, Configuring, and Starting GDS</td>
<td>265</td>
</tr>
<tr>
<td>5.2 Stopping GDS</td>
<td>269</td>
</tr>
<tr>
<td>5.3 Example</td>
<td>270</td>
</tr>
<tr>
<td>5.4 gds Command Introduction</td>
<td>273</td>
</tr>
</tbody>
</table>
6 Migration Tool - SQL Syntax Migration Tool

5.5 gds_ctl.py Command Introduction

6.1 Overview

6.2 Supported Keywords and Features

6.2.1 Overview

6.2.2 Teradata Keywords and Features

6.2.3 Oracle Keywords and Features

6.2.4 Oracle (Beta) Keywords and Features

6.3 System Requirements

6.4 Constraints and Limitations

6.5 Installing Migration Tool

6.6 Configuring the Migration Tool

6.6.1 Overview

6.6.2 Migration Tool Configuration

6.6.3 Teradata SQL Configuration

6.6.4 Oracle SQL Configuration

6.6.5 Oracle (Beta) SQL Configuration

6.6.6 Teradata Perl Configuration

6.6.7 Custom DB Script Configuration

6.7 Migration Process

6.7.1 Overview

6.7.2 Prerequisites

6.7.3 Preparations

6.7.4 Executing Migration Tool

6.7.5 Viewing Output Files and Logs

6.7.6 Troubleshooting

6.7.7 Teradata SQL Migration

6.8 Oracle SQL Migration

6.9 Oracle (Beta) SQL Migration

6.10 Teradata Perl Migration

6.11 SQL Formatter

6.12 Teradata Syntax Migration

6.12.1 Overview

6.12.2 Schema Objects

6.12.2.1 Overview

6.12.2.2 Table Migration

6.12.2.3 CHARACTER SET and CASESPECIFIC

6.12.2.4 Index Migration

6.12.2.5 View Migration

6.12.2.6 Collect Statistics Migration

6.12.2.7 ACCESS LOCK

6.12.2.8 DBC Columns
6.12.2.9 DBC Tables............................................................................................................................................................................ 403
6.12.2.10 DBC Indices........................................................................................................................................................................ 403
6.12.3 DML.................................................................................................................................................................................... 404
6.12.3.1 Overview..........................................................................................................................................................................404
6.12.3.2 INSERT.............................................................................................................................................................................. 404
6.12.3.3 SELECT............................................................................................................................................................................. 404
6.12.3.4 UPDATE........................................................................................................................................................................... 411
6.12.3.5 DELETE........................................................................................................................................................................... 411
6.12.3.6 MERGE........................................................................................................................................................................... 412
6.12.3.7 NAMED........................................................................................................................................................................... 413
6.12.3.8 ANALYZE........................................................................................................................................................................ 414
6.12.3.9 ACTIVITYCOUNT.......................................................................................................................................................... 415
6.12.3.10 TIMESTAMP................................................................................................................................................................. 416
6.12.3.11 LOGOFF QUIT.............................................................................................................................................................. 416
6.12.3.12 COLLECT STATISTICS............................................................................................................................................... 416
6.12.3.13 IF ERRORCODE and .QUIT........................................................................................................................................ 417
6.12.4 Query Migration Operators............................................................................................................................................... 417
6.12.4.1 Overview...................................................................................................................................................................... 417
6.12.4.2 QUALIFY........................................................................................................................................................................ 417
6.12.4.3 ALIAS............................................................................................................................................................................. 420
6.12.4.4 FORMAT AND CAST.................................................................................................................................................... 421
6.12.4.5 Short Keys Migration.................................................................................................................................................... 423
6.12.4.6 Object Names Starting With $......................................................................................................................................... 425
6.12.5 Query Optimization Operators........................................................................................................................................... 427
6.12.5.1 Overview...................................................................................................................................................................... 427
6.12.5.2 IN and NOT IN Conversion........................................................................................................................................... 428
6.12.6 System Functions and Operators....................................................................................................................................... 429
6.12.6.1 Overview...................................................................................................................................................................... 429
6.12.6.2 Analytical Functions.................................................................................................................................................... 429
6.12.6.3 Window Functions.......................................................................................................................................................... 430
6.12.6.4 Comparison and List Operators..................................................................................................................................... 433
6.12.6.5 Table Operators.......................................................................................................................................................... 435
6.12.7 Math Functions................................................................................................................................................................. 435
6.12.8 String Functions................................................................................................................................................................. 436
6.12.9 Date and Time Functions..................................................................................................................................................... 437
6.12.10 Type Casting and Formatting.......................................................................................................................................... 439
6.13 Oracle Syntax Migration....................................................................................................................................................... 445
6.13.1 Overview............................................................................................................................................................................. 445
6.13.2 Schema Objects................................................................................................................................................................. 445
6.13.2.1 Overview...................................................................................................................................................................... 445
6.13.2.2 Tables........................................................................................................................................................................... 445
6.13.2.3 Temporary Tables.......................................................................................................................................................... 468
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.13.9.15</td>
<td>AUTONOMOUS</td>
<td>540</td>
</tr>
<tr>
<td>6.13.9.16</td>
<td>Procedure Call</td>
<td>540</td>
</tr>
<tr>
<td>6.13.10</td>
<td>PL/SQL Collections (Using User-Defined Types)</td>
<td>544</td>
</tr>
<tr>
<td>6.13.11</td>
<td>PL/SQL Packages</td>
<td>550</td>
</tr>
<tr>
<td>6.13.11.1</td>
<td>Overview</td>
<td>550</td>
</tr>
<tr>
<td>6.13.11.2</td>
<td>Packages</td>
<td>550</td>
</tr>
<tr>
<td>6.13.11.3</td>
<td>Package Variables</td>
<td>561</td>
</tr>
<tr>
<td>6.13.11.4</td>
<td>Package Split</td>
<td>578</td>
</tr>
<tr>
<td>6.13.11.5</td>
<td>REF CURSOR</td>
<td>581</td>
</tr>
<tr>
<td>6.13.11.6</td>
<td>VARRAY</td>
<td>582</td>
</tr>
<tr>
<td>6.13.11.7</td>
<td>Creating a Schema for Package</td>
<td>583</td>
</tr>
<tr>
<td>6.13.11.8</td>
<td>Grant Execute</td>
<td>583</td>
</tr>
<tr>
<td>6.13.11.9</td>
<td>Package Name List</td>
<td>585</td>
</tr>
<tr>
<td>6.13.11.10</td>
<td>Data Type</td>
<td>585</td>
</tr>
<tr>
<td>6.13.12</td>
<td>Chinese Character Support</td>
<td>587</td>
</tr>
<tr>
<td>6.14</td>
<td>Oracle (Beta) Syntax Migration</td>
<td>587</td>
</tr>
<tr>
<td>6.14.1</td>
<td>Overview</td>
<td>588</td>
</tr>
<tr>
<td>6.14.2</td>
<td>Schema Objects</td>
<td>588</td>
</tr>
<tr>
<td>6.14.2.1</td>
<td>Overview</td>
<td>588</td>
</tr>
<tr>
<td>6.14.2.2</td>
<td>Tables</td>
<td>588</td>
</tr>
<tr>
<td>6.14.2.3</td>
<td>Temporary Tables</td>
<td>596</td>
</tr>
<tr>
<td>6.14.2.4</td>
<td>Indexes</td>
<td>597</td>
</tr>
<tr>
<td>6.14.2.5</td>
<td>Sequences</td>
<td>600</td>
</tr>
<tr>
<td>6.14.2.6</td>
<td>Purge</td>
<td>602</td>
</tr>
<tr>
<td>6.14.2.7</td>
<td>Database Keywords</td>
<td>602</td>
</tr>
<tr>
<td>6.14.2.8</td>
<td>Synonym</td>
<td>602</td>
</tr>
<tr>
<td>6.14.2.9</td>
<td>Gauss Keyword</td>
<td>603</td>
</tr>
<tr>
<td>6.14.3</td>
<td>DML</td>
<td>604</td>
</tr>
<tr>
<td>6.14.3.1</td>
<td>Overview</td>
<td>604</td>
</tr>
<tr>
<td>6.14.3.2</td>
<td>INSERT</td>
<td>604</td>
</tr>
<tr>
<td>6.14.4</td>
<td>Pseudo Columns</td>
<td>609</td>
</tr>
<tr>
<td>6.14.4.1</td>
<td>Overview</td>
<td>610</td>
</tr>
<tr>
<td>6.14.4.2</td>
<td>ROWNUM</td>
<td>610</td>
</tr>
<tr>
<td>6.14.4.3</td>
<td>ROWID</td>
<td>612</td>
</tr>
<tr>
<td>6.14.5</td>
<td>OUTER JOIN</td>
<td>612</td>
</tr>
<tr>
<td>6.14.6</td>
<td>System Functions</td>
<td>615</td>
</tr>
<tr>
<td>6.14.6.1</td>
<td>Overview</td>
<td>615</td>
</tr>
<tr>
<td>6.14.6.2</td>
<td>Date Functions</td>
<td>615</td>
</tr>
<tr>
<td>6.14.6.3</td>
<td>Other Functions</td>
<td>617</td>
</tr>
<tr>
<td>6.14.7</td>
<td>PL/SQL</td>
<td>621</td>
</tr>
<tr>
<td>6.14.7.1</td>
<td>Overview</td>
<td>621</td>
</tr>
<tr>
<td>6.14.7.2</td>
<td>END</td>
<td>621</td>
</tr>
</tbody>
</table>
This document describes the tools used in Data Warehouse Service (DWS). **Table 1-1** lists the tools and their supported versions.

**Table 1-1 Tools**

<table>
<thead>
<tr>
<th>Tool</th>
<th>Version</th>
<th>Description</th>
<th>Supported Cluster Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>gsql</td>
<td>V300R00 2C00</td>
<td>A command line SQL client tool running on the Linux OS. It is used to connect to the database in a data warehouse cluster and operate and maintain the database.</td>
<td>DWS 1.0.x or later</td>
</tr>
<tr>
<td>Data Studio</td>
<td>6.5.1</td>
<td>A client tool used to connect to a database. It provides a GUI for managing databases and objects; editing, running, and debugging SQL scripts; and viewing execution plans. Data Studio can run on a 32-bit or 64-bit Windows OS. You can use it after decompression without installation.</td>
<td>DWS 1.2.x or later</td>
</tr>
<tr>
<td>GDS</td>
<td>V300R00 2C00</td>
<td>A command line tool running on the Linux OS. It works with the foreign table mechanism to quickly import and export data. The GDS tool package needs to be installed on the server where the data source file is located. This server is called the data server or the GDS server.</td>
<td>DWS 1.1.x or later</td>
</tr>
<tr>
<td>Tool</td>
<td>Version</td>
<td>Description</td>
<td>Supported Cluster Version</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>Migration Tool</td>
<td>6.5.1</td>
<td>A command line tool used for migrating SQL scripts from Teradata or Oracle to DWS to rebuild a database on DWS. Migration Tool runs on the Linux OS. You can use it after decompression without installation.</td>
<td>DWS 1.3.x or later</td>
</tr>
<tr>
<td>Schema Convertor Tool</td>
<td>1.0</td>
<td>Schema Convertor Tool is a command line tool running on the Linux or Windows OS and is used to migrate and transform the SQL scripts in MySQL databases to SQL scripts applicable to DWS to rebuild the databases on DWS. You can use it after decompressing the software package without installation.</td>
<td>DWS 1.3.x or later</td>
</tr>
</tbody>
</table>
2 Downloading Related Tools

gsql CLI Client (Containing the GDS Package)

Table 2-1 gsql download links

<table>
<thead>
<tr>
<th>Applicable Operating System</th>
<th>Download Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>RHEL 6.4/6.5/6.6/6.7/7.1/7.2 CentOS 6.4/6.5/6.6/6.7 EulerOS 2.0 SP2</td>
<td>dws_client_1.5.x_redhat_x64.zip</td>
</tr>
<tr>
<td></td>
<td>dws_client_1.3.x_redhat_x64.tar.gz</td>
</tr>
<tr>
<td></td>
<td>dws_client_1.2.x_redhat_x64.tar.gz</td>
</tr>
<tr>
<td>SLES11 SP1/SP2/SP3/SP4</td>
<td>dws_client_1.5.x_suse_x64.zip</td>
</tr>
<tr>
<td></td>
<td>dws_client_1.3.x_suse_x64.tar.gz</td>
</tr>
<tr>
<td></td>
<td>dws_client_1.2.x_suse_x64.tar.gz</td>
</tr>
</tbody>
</table>

Data Studio GUI Client

Table 2-2 Data Studio download links

<table>
<thead>
<tr>
<th>Applicable Operating System</th>
<th>Download Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Windows x64</td>
<td>Data_Studio_1.5.x_64.zip</td>
</tr>
<tr>
<td></td>
<td>Data_Studio_1.3.x_64.zip</td>
</tr>
<tr>
<td></td>
<td>Data_Studio_1.2.x_64.zip</td>
</tr>
<tr>
<td>Microsoft Windows x86</td>
<td>Data_Studio_1.5.x_32.zip</td>
</tr>
<tr>
<td></td>
<td>Data_Studio_1.3.x_32.zip</td>
</tr>
<tr>
<td></td>
<td>Data_Studio_1.2.x_32.zip</td>
</tr>
</tbody>
</table>
Migration Tool

Table 2-3 Migration Tool download links

<table>
<thead>
<tr>
<th>Applicable Operating System</th>
<th>Download Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>For details, see Migration Tool - SQL Syntax Migration Tool &gt; About Migration Tool &gt; System Requirements.</td>
<td>MigrationTool.zip</td>
</tr>
</tbody>
</table>

Convertor Migration Tool

Table 2-4 Convertor download links

<table>
<thead>
<tr>
<th>Applicable Operating System</th>
<th>Download Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>For details, see Schema Convertor Tool: SQL Syntax Migration Tool &gt; Introduction to Schema Convertor Tool &gt; System Requirements.</td>
<td>dws-convertor-1.0-SNAPSHOT.zip</td>
</tr>
</tbody>
</table>

SSL Certificate Download Link

dws_ssl_cert.zip

DWS JDBC Driver

Table 2-5 DWS JDBC driver download links

<table>
<thead>
<tr>
<th>Applicable Operating System</th>
<th>Download Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Windows x64/x86</td>
<td>dws_1.5.x_jdbc_driver.zip</td>
</tr>
<tr>
<td>Linux x64/x86</td>
<td>dws_1.3.x_jdbc_driver.tar.gz</td>
</tr>
<tr>
<td>Applicable to JDK 1.6 or later</td>
<td>dws_1.2.x_jdbc_driver.tar.gz</td>
</tr>
</tbody>
</table>
## DWS ODBC Driver

### Table 2-6 DWS ODBC driver download links

<table>
<thead>
<tr>
<th>Applicable Operating System</th>
<th>Download Link</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Microsoft Windows x64/x86:</strong></td>
<td></td>
</tr>
<tr>
<td>● Applicable to Windows Server 2008 or later</td>
<td>dws_1.5.x_odbc_driver_for_windows.zip</td>
</tr>
<tr>
<td>● Applicable to Windows 7 or later</td>
<td>dws_1.3.x_odbc_driver_for_windows.tar.gz</td>
</tr>
<tr>
<td>● Applicable to Windows 7 or later</td>
<td>dws_1.2.x_odbc_driver_for_windows.tar.gz</td>
</tr>
<tr>
<td><strong>Linux x64:</strong></td>
<td></td>
</tr>
<tr>
<td>● RHEL 6.4/6.5/6.6/6.7/7.1/7.2</td>
<td>dws_1.5.x_odbc_driver_for_linux.zip</td>
</tr>
<tr>
<td>● CentOS 6.4/6.5/6.6/6.7</td>
<td>dws_1.3.x_odbc_driver_for_linux.tar.gz</td>
</tr>
<tr>
<td>● EulerOS 2.0 SP2; SLES11 SP1/SP2/SP3/SP4</td>
<td>dws_1.2.x_odbc_driver_for_linux.tar.gz</td>
</tr>
</tbody>
</table>
3.1 gsql Overview

Basic Functions

- **Connect to the database**: Use the `gsql` client to remotely connect to the DWS database.

  **NOTE**
  
  If the `gsql` client is used to connect to a database, the connection timeout period will be 5 minutes. If the database has not correctly set up a connection and authenticated the identity of the client within this period, `gsql` will time out and exit.
  
  To resolve this problem, see Troubleshooting.

- **Run SQL statements**: Interactively entered SQL statements and specified SQL statements in a file can be run.

- **Run meta-commands**: Meta-commands help the administrator view database object information, query cache information, format SQL output, and connect to a new database. For details about meta-commands, see Meta-Command Reference.

Advanced Features

*Table 3-1* lists the advanced features of `gsql`. 
Table 3-1 Advanced features of gsql

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>gsql provides a variable feature that is similar to the shell command of Linux. The following \set meta-command of gsql can be used to set a variable: \set varname value&lt;br&gt;<strong>To delete a variable, run the following command:</strong> \unset varname&lt;br&gt;&lt;br&gt;<strong>NOTE</strong>&lt;br&gt;• A variable is a simple name-value pair, where the value can be any characters in any length.&lt;br&gt;• Variable names must consist of case-sensitive letters (including non-Latin letters), digits, and underscores(_).&lt;br&gt;• If the \set varname meta-command (without the second parameter) is used, the variable is set without a value specified.&lt;br&gt;• If the \set meta-command without parameters is used, values of all variables are displayed.&lt;br&gt;For details about variable examples and descriptions, see Variables.</td>
</tr>
<tr>
<td>SQL substitution</td>
<td>Common SQL statements can be set to variables using the variable feature of gsql to simplify operations.&lt;br&gt;For details about SQL substitution examples and descriptions, see SQL Substitution.</td>
</tr>
<tr>
<td>Customized prompt</td>
<td>Prompts of gsql can be customized. Prompts can be modified by changing the reserved variables of gsql: PROMPT1, PROMPT2, and PROMPT3.&lt;br&gt;These variables can be set to customized values or the values predefined by gsql. For details, see Prompt.</td>
</tr>
<tr>
<td>Client operation history record</td>
<td>gsql records client operation history. This function is enabled by specifying the -r parameter when a client is connected. The number of historical records can be set using the \set command. For example, \set HISTSIZE 50 indicates that the number of historical records is set to 50. \set HISTSIZE 0 indicates that the operation history is not recorded.&lt;br&gt;&lt;br&gt;<strong>NOTE</strong>&lt;br&gt;• The default number of historical records is 32. The maximum number of historical records is 500. If interactively entered SQL statements contain Chinese characters, only the UTF-8 encoding environment is supported.&lt;br&gt;• For security reasons, the records containing sensitive words, such as PASSWORD and IDENTIFIED, are regarded sensitive and not recorded in historical information. This indicates that you cannot view these records in command output histories.</td>
</tr>
</tbody>
</table>

- **Variable**<br>To set a variable, run the \set meta-command of gsql. For example, to set variable foo to bar, run the following command:<br>\set foo bar
To quote the value of a variable, add a colon (:) before the variable. For example, to view the value of variable `foo`, run the following command:
```
\echo :foo
bar
```

The variable quotation method is suitable for regular SQL statements and meta-commands.

`gsql` pre-defines some special variables and plans the values of these variables. To ensure compatibility with later versions, do not use these variables for other purposes. For details about special variables, see **Table 3-2**.

**NOTE**

- All the special variables consist of uppercase letters, digits, and underscores (_).
- To view the default value of a special variable, run the `\echo :varname` meta-command, for example, `\echo :DBNAME`.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Setting Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBNAME</td>
<td><code>\set DBNAME dbname</code></td>
<td>Specifies the name of a connected database. This variable is set again when a database is connected.</td>
</tr>
</tbody>
</table>
| ECHO          | `\set ECHO all | queries`                        | - If this variable is set to **all**, only the query information is displayed. This has the same effect as specifying the `-a` parameter when `gsql` is used to connect to a database.  
    - If this variable is set to **queries**, the command line and query information are displayed. This has the same effect as specifying the `-e` parameter when `gsql` is used to connect to a database. |
| ECHO_HID DEN   | `\set ECHO_HIDDEN on | off | noexec`                         | When a meta-command (such as `\dg`) is used to query database information, the value of this variable determines the query behavior.  
    - If this variable is set to **on**, the query statements that are called by the meta-command are displayed, and then the query result is displayed. This has the same effect as specifying the `-E` parameter when `gsql` is used to connect to a database.  
    - If this variable is set to **off**, only the query result is displayed.  
    - If this variable is set to **noexec**, only the query information is displayed, and the query is not run. |
<table>
<thead>
<tr>
<th>Variable</th>
<th>Setting Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoding</td>
<td><code>set ENCODING encoding</code></td>
<td>Specifies the character set encoding of the current client. Workbench will use the current client's character set encoding. If the encoding is specified, Workbench will use the specified encoding.</td>
</tr>
</tbody>
</table>
| Fetch_count    | `set FETCH_COUNT variable`              | - If the value is an integer greater than 0, for example, `n`, `n` lines will be selected from the result set to the cache and displayed on the screen when the `SELECT` statement is run.  
- If this variable is not set or set to a value less than or equal to 0, all results are selected at a time to the cache when the `SELECT` statement is run. |
| Hist控制       | `set HISTCONTROL {ignore space | ignoredups | ignore both | none}`                           | - `ignore space`: A line started with a space is not written to the historical record.  
- `ignore duplicated`: A line that exists in the historical record is not written to the historical record.  
- `ignore both`, `none`, or other values: All the lines read in interaction mode are saved in the historical record.  
NOTE: `none` indicates that HISTCONTROL is not set. |
| Histfile       | `set HISTFILE filename`                 | Specifies the file for storing historical records. The default value is `~/.bash_history`.                                                                                                            |
| Histsize       | `set HISTSIZE size`                     | Specifies the number of commands in the history command. The default value is 500.                                                                                                                     |
| Host           | `set HOST hostname`                     | Specifies the name of a connected host.                                                                                                                                                                  |
| Ignoreeof      | `set IGNOREEOF variable`               | - If this variable is set to a number, for example, `10`, the first nine EOF characters (generally `Ctrl+c`) entered in `gsql` are neglected and the `gsql` program exits when the tenth `Ctrl+c` is entered.  
- If this variable is set to a non-numeric value, the default value is `10`.  
- If this variable is deleted, `gsql` exits when an EOF is entered. |
<table>
<thead>
<tr>
<th>Variable</th>
<th>Setting Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LASTOID</td>
<td>\set LASTOID oid</td>
<td>Specifies the last OID, which is the value returned by an <strong>INSERT</strong> or <strong>lo_import</strong> command. This variable is valid only before the output of the next SQL statement is displayed.</td>
</tr>
<tr>
<td>ON_ERROR_ROLLBACK</td>
<td>\set ON_ERROR_ROLLBACK on</td>
<td>- If the value is <strong>on</strong>, an error that may occur in a statement in a transaction block is ignored and the transaction continues.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- If the value is <strong>interactive</strong>, the error is ignored only in an interactive session.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- If the value is <strong>off</strong> (the default value), the error triggers the rollback of the transaction block. In <strong>on_error_rollback-on</strong> mode, a <strong>SAVEPOINT</strong> is set before each statement of a transaction block, and an error triggers the rollback of the transaction block.</td>
</tr>
<tr>
<td>ON_ERROR_STOP</td>
<td>\set ON_ERROR_STOP on</td>
<td>- <strong>on</strong>: specifies that the execution stops if an error occurs. In interactive mode, <strong>gsql</strong> returns the output of executed commands immediately.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>off</strong> (default value): specifies that an error, if occurring during the execution, is ignored, and the execution continues.</td>
</tr>
<tr>
<td>PORT</td>
<td>\set PORT port</td>
<td>Specifies the port number of a connected database.</td>
</tr>
<tr>
<td>USER</td>
<td>\set USER username</td>
<td>Specifies the connected database user.</td>
</tr>
<tr>
<td>VERBOSITY</td>
<td>\set VERBOSITY terse</td>
<td>This variable can be set to <strong>terse</strong>, <strong>default</strong>, or <strong>verbose</strong> to control redundant lines of error reports.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>terse</strong>: Only critical and major error texts and text locations are returned (which is suitable for single-line error information).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>default</strong>: Critical and major error texts and text locations, error details, and error messages (possibly involving multiple lines) are all returned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>verbose</strong>: All error information is returned.</td>
</tr>
</tbody>
</table>
• SQL substitution

`gsql`, like a parameter of a meta-command, provides a key feature that enables you to substitute a standard SQL statement for a `gsql` variable. `gsql` also provides a new alias or identifier for the variable. To replace the value of a variable using the SQL substitution method, add a colon (:) in front of the variable. For example:

```sql
\set foo 'HR.areaS'
select * from :foo;
```

<table>
<thead>
<tr>
<th>area_id</th>
<th>area_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Middle East and Africa</td>
</tr>
<tr>
<td>3</td>
<td>Asia</td>
</tr>
<tr>
<td>1</td>
<td>Europe</td>
</tr>
<tr>
<td>2</td>
<td>Americas</td>
</tr>
</tbody>
</table>

(4 rows)

The above command queries the `HR.areaS` table.

**NOTICE**

The value of a variable is copied character by character, and even an asymmetric quote mark or backslash (\) is copied. Therefore, the input content must be meaningful.

• Prompt

The `gsql` prompt can be set using the three variables in Table 3-3. These variables consist of characters and special escape characters.

**Table 3-3 Prompt variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| PROMPT1  | Specifies the normal prompt used when `gsql` requests a new command. The default value of `PROMPT1` is: %/%R%# | `PROMPT1` can be used to change the prompt.  
- Change the prompt to `[local]`: `postgres=> \set PROMPT1 %M [local:/tmp/gaussdba_mppdb]`  
- Change the prompt to `name`: `postgres=> \set PROMPT1 name name`  
- Change the prompt to `=`: `postgres=> \set PROMPT1 %R =` |
| PROMPT2  | Specifies the prompt displayed when more command input is expected. For example, it is expected if a command is not terminated with a semicolon (;) or a quote (") is not closed. | `PROMPT2` can be used to display the prompt:  
`postgres=> \set PROMPT2 TEST select * from HR.areaS TEST;`  
<table>
<thead>
<tr>
<th>area_id</th>
<th>area_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Europe</td>
</tr>
<tr>
<td>2</td>
<td>Americas</td>
</tr>
<tr>
<td>4</td>
<td>Middle East and Africa</td>
</tr>
<tr>
<td>3</td>
<td>Asia</td>
</tr>
</tbody>
</table>

(4 rows)
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| PROMPT3    | Specifies the prompt displayed when the COPY statement (such as COPY FROM STDIN) is run and data input is expected.                                                                                      | PROMPT3 can be used to display the COPY prompt. \set PROMPT3 '>>>>'
copy HR.areaS from STDIN;
Enter data to be copied followed by a newline. End with a backslash and a period on a line by itself.
>>>>1 aa
>>>>2 bb
>>>>.                                                                 |

The value of the selected prompt variable is printed literally. However, a value containing a percent sign (%) is replaced by the predefined contents depending on the character following the percent sign (%). For details about the defined substitutions, see Table 3-4.

Table 3-4 Defined substitutions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%M</td>
<td>Specifies the full host name (with domain name). The full name is [local] if the connection is over a Unix domain socket, or [local:/dir/name] if the Unix domain socket is not at the compiled default location.</td>
</tr>
<tr>
<td>%m</td>
<td>Specifies the host name truncated at the first dot. It is [local] if the connection is over a Unix domain socket.</td>
</tr>
<tr>
<td>%&gt;</td>
<td>Specifies the number of the port that the host is listening on.</td>
</tr>
<tr>
<td>%n</td>
<td>Specifies the database session user name.</td>
</tr>
<tr>
<td>%/</td>
<td>Specifies the name of the current database.</td>
</tr>
<tr>
<td>%~</td>
<td>Is similar to %/. However, the output is tilde (~) if the database is your default database.</td>
</tr>
<tr>
<td>%#</td>
<td>Uses # if the session user is the database administrator. Otherwise, uses &gt;.</td>
</tr>
</tbody>
</table>
| %R     | • Normally uses = for PROMPT1, but ^ in single-line mode and ! if the session is disconnected from the database (which may occur if `connect` fails).
  • Is replaced with a hyphen (-), an asterisk (*), a single or double quotation mark, or a dollar sign ($) for PROMPT2, depending on whether `gsql` expects more input because the query is inside a /*...*/ comment or inside a quoted or dollar-escaped string. |
Symbol | Description
--- | ---
%\*x | Specifies the transaction status.  
- An empty string when it is not in a transaction block  
- An asterisk (*) when it is in a transaction block  
- An exclamation mark (!) when it is in a failed transaction block  
- A question mark (?) when the transaction status is indeterminate (for example, indeterminate due to no connections).

%\%digits | Is replaced with the character with the specified byte.

%\%:name | Specifies the value of the name variable of gsq1.

%\%comma | Specifies command output, similar to ordinary "back-tick" ("^") substitution.

%\%[ ... %] | Prompts can contain terminal control characters which, for example, change the color, background, or style of the prompt text, or change the title of the terminal window. For example:

```sql
postgres=> \set PROMPT1 '%[\033[1;33;40m%]n@%/%R% [\033[0m%]%#'
```

The output is a boldfaced (1;) yellow-on-black (33;40) prompt on VT100-compatible, color-capable terminals.

**Environment Variables**

Table 3-5 Environment variables related to gsq1

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLUMNS</td>
<td>If \set columns is set to 0, this parameter controls the width of the wrapped format. This width determines whether the width output mode is changed to a vertical bar format in automatic expansion mode.</td>
</tr>
</tbody>
</table>
| PAGER        | If the query result cannot be displayed within one page, the query result will be redirected to the command. You can use the \pset command to disable the pager. Typically, the more or less command is used for viewing the query result page by page. The default value is platform-associated.  
**NOTE**  
Display of the less command is affected by the LC_CTYPE environmental variable. |
| PSQL_EDITOR  | The \e and \ef commands use the editor specified by the environment variables. Variables are checked according to the list sequence. The default editor on Unix is vi. |
| EDITOR       | |
### 3.2 Instruction

**Downloading and Installing `gsql` and Using It to Connect to the Cluster's Database**

For details about how to download and install `gsql`, and connect it to the cluster's database, see section *Using the `gsql` CLI Client to Connect to a Cluster* in the *Data Warehouse Cluster Management Guide*.

**Example**

The example shows how to spread a command over several lines of input. Pay attention to prompt changes:

```sql
postgres=# CREATE TABLE HR.areaS(
    area_ID   NUMBER,
    area_NAME VARCHAR2(25)
) tablespace EXAMPLE;

CREATE TABLE

View the table definition.

```
\d HR.areaS
```

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Modifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>area_id</td>
<td>numeric</td>
<td>not null</td>
</tr>
<tr>
<td>area_name</td>
<td>character varying(25)</td>
<td></td>
</tr>
</tbody>
</table>

Insert four lines of data into `HR.areaS`.

```
INSERT INTO HR.areaS (area_ID, area_NAME) VALUES (1, 'Wood');
INSERT 0 1
INSERT INTO HR.areaS (area_ID, area_NAME) VALUES (2, 'Lake');
INSERT 0 1
```

---

**Table**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VISUAL</td>
<td></td>
</tr>
<tr>
<td><code>PSQL_EDITOR_LINENUMBER_ARG</code></td>
<td>When the <code>\e</code> or <code>\ef</code> command is used with a line number parameter, this variable specifies the command-line parameter used to pass the starting line number to the editor. For editors, such as Emacs or vi, this is a plus sign. A space is added behind the value of the variable if whitespace is required between the option name and the line number. For example: <code>PSQL_EDITOR_LINENUMBER_ARG = '+'</code> <code>PSQL_EDITOR_LINENUMBER_ARG='--line '</code> A plus sign (+) is used by default on Unix.</td>
</tr>
<tr>
<td><code>PSQLRC</code></td>
<td>Specifies the location of the user's <code>.gsqlrc</code> file.</td>
</tr>
<tr>
<td><code>SHELL</code></td>
<td>Has the same effect as the <code>\!</code> command.</td>
</tr>
<tr>
<td><code>TMPDIR</code></td>
<td>Specifies the directory for storing temporary files. The default value is <code>/tmp</code>.</td>
</tr>
</tbody>
</table>
INSERT INTO HR.areaS (area_ID, area_NAME) VALUES (3, 'Desert');
INSERT INTO HR.areaS (area_ID, area_NAME) VALUES (4, 'Iron');

Change the prompt.
```
\set PROMPT1 '%n@%m %~%R%#'
dbadmin@[local] postgres=#
```

View the table.
```
dbadmin@[local] postgres=#SELECT * FROM HR.areaS;
area_id | area_name
---------+------------------------
1 | Wood
4 | Iron
2 | Lake
3 | Desert
(4 rows)
```

Run the \texttt{\textbackslash pset} command to display the table in different ways.
```
dbadmin@[local] postgres=#\texttt{\textbackslash pset border 2}
Border style is 2.
dbadmin@[local] postgres=#SELECT * FROM HR.areaS;
+---------+------------------------+
| area_id | area_name              |
+---------+------------------------+
| 1 | Wood                 |
| 2 | Lake              |
| 3 | Desert                   |
| 4 | Iron |
+---------+------------------------+
(4 rows)
dbadmin@[local] postgres=#\texttt{\textbackslash pset border 0}
Border style is 0.
dbadmin@[local] postgres=#SELECT * FROM HR.areaS;
area_id | area_name
---------+------------------------
1 Wood
2 Lake
3 Desert
4 Iron
(4 rows)
```

Use the meta-command.
```
dbadmin@[local] postgres=#\texttt{\textbackslash a \t \x}
Output format is unaligned.
Showing only tuples.
Expanded display is on.
dbadmin@[local] postgres=#SELECT * FROM HR.areaS;
area_id | area_name|Lake
area_id | Wood
area_id | Iron
area_id | Desert
(4 rows)
```

---

**Data Warehouse Service Tool Guide**

Issue 03 (2019-08-08)  Copyright © Huawei Technologies Co., Ltd.  15
3.3 Online Help

Procedure

- When a database is being connected, run the following commands to obtain the help information:
  
gsql --help

  The following information is displayed:

  ......
  Usage:
gsql [OPTION]... [DBNAME [USERNAME]]

  General options:
  -c, --command=COMMAND run only single command (SQL or internal) and exit
  -d, --dbname=DBNAME database name to connect to (default: "postgres")
  -f, --file=FILENAME execute commands from file, then exit
  ......

- After the database is connected, run the following commands to obtain the help information:
  
help

  The following information is displayed:

  You are using gsql, the command-line interface to gaussdb.
  Type: \copyright for distribution terms
  \h for help with SQL commands
  \? for help with gsql commands
  \g or terminate with semicolon to execute query
  \q to quit

Task Example

Step 1  Run the following command to connect to the database:

  gsql -d postgres -p 8000

  postgres is the name of the database, and 8000 is the port number of the CN.

  If information similar to the following is displayed, the connection succeeds:

  gsql ((GaussDB Kernel V300R002C00 build 50dc16a6) compiled at 2019-01-29 05:49:21 commit 1071 last
  mr 1373)
  Non-SSL connection (SSL connection is recommended when requiring high-security)
  Type "help" for help.

  postgres=#

Step 2  View the gsql help information. For details about the commands, see Table 3-6.

Table 3-6 gsql online help

<table>
<thead>
<tr>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>View copyright</td>
<td>\copyright</td>
</tr>
</tbody>
</table>
  information.          |          |
**Description** | **Example**
--- | ---
View the help information about SQL statements supported by DWS. | View the help information about SQL statements supported by DWS. For example, view all SQL statements supported by DWS.

```
\h
```
Available help:
- ABORT
- ALTER AGGREGATE
- ALTER APP WORKLOAD GROUP
- ...

For example, view parameters of the `CREATE DATABASE` command:
```
\help CREATE DATABASE
Command: CREATE DATABASE
Description: create a new database
Syntax:
CREATE DATABASE database_name
[ [ WITH ] [ { OWNER = user_name | 
 TEMPLATE = template } ]
[ ENCODING = encoding ]
[ LC_COLLATE = lc_collate ]
[ LC_CTYPE = lc_ctype ]
[ DBCOMPATIBILITY = compatibility_type ]
[ TABLESPACE = tablespace_name ]
[ CONNECTION LIMIT = connlimit ] } ];
```

View help information about gsql commands. | For example, view commands supported by `gsql`.

```
\?
```
General:
- `\copyright` show PostgreSQL usage and distribution terms
- `\g [FILE] or ;` execute query (and send results to file or `|pipe`)
- `\h [NAME]` help on syntax of SQL commands, * for all commands
- `\q` quit gsql
- ...

### End

### 3.4 Command Reference

For details about gsql parameters, see Table 3-7, Table 3-8, Table 3-9, and Table 3-10.

**Table 3-7 Common parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>-c, --command=COMMAND</td>
<td>Specifies that gsql is to run a string command and then exit.</td>
<td>-</td>
</tr>
<tr>
<td>-d, --dbname=DBNAME</td>
<td>Specifies the name of the database to be connected.</td>
<td>A character string.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>-f, --file=FILENAME</td>
<td>Specifies that files are used as the command source instead of interactively-entered commands. After the files are processed, gsql exits. If FILENAME is - (hyphen), then standard input is read.</td>
<td>An absolute path or relative path that meets the OS path naming convention</td>
</tr>
<tr>
<td>-l, --list</td>
<td>Lists all available databases and then exits.</td>
<td>-</td>
</tr>
<tr>
<td>-v, --set, --variable=NAME=VALUE</td>
<td>Sets the gsql variable NAME to VALUE. For details about variable examples and descriptions, see Variables.</td>
<td>-</td>
</tr>
<tr>
<td>-X, --no-gsqlrc</td>
<td>Does not read the startup file (neither the system-wide gsqlrc file nor the user's ~/.gsqlrc file). [NOTE] The startup file is ~/.gsqlrc by default or it can be specified by the environment variable PSQLRC.</td>
<td>-</td>
</tr>
<tr>
<td>-1 (&quot;one&quot;), --single-transaction</td>
<td>When gsql uses the -f parameter to execute a script, START TRANSACTION and COMMIT are added to the start and end of the script, respectively, so that the script is executed as one transaction. This ensures that the script is executed successfully. If the script cannot be executed, the script is invalid. [NOTE] If the script has used START TRANSACTION, COMMIT, and ROLLBACK, this parameter is invalid.</td>
<td>-</td>
</tr>
<tr>
<td>-?, --help</td>
<td>Displays help information about gsql command line parameters, and exits.</td>
<td>-</td>
</tr>
<tr>
<td>-V, --version</td>
<td>Prints the gsql version and exits.</td>
<td>-</td>
</tr>
</tbody>
</table>

### Table 3-8 Input and output parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>-a, --echo-all</td>
<td>Prints all input lines to standard output as they are read. [CAUTION] When this parameter is used in some SQL statements, sensitive information, such as user passwords, may be disclosed. Use this parameter with caution.</td>
<td>-</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>-e, --echo-queries</td>
<td>Copies all SQL statements sent to the server to standard output as well.</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>CAUTION</strong> When this parameter is used in some SQL statements, sensitive information, such as user passwords, may be disclosed. Use this parameter with caution.</td>
<td></td>
</tr>
<tr>
<td>-E, --echo-hidden</td>
<td>Echoes the actual queries generated by <code>\d</code> and other backslash commands.</td>
<td>-</td>
</tr>
<tr>
<td>-k, --with-key=KEY</td>
<td>Uses <code>gsql</code> to decrypt imported encrypted files.</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>NOTICE</strong> For key characters, such as the single quotation mark (<code>) or double quotation mark (&quot;), in shell commands, Linux shell checks whether the input single quotation mark (</code>) or double quotation mark (&quot;) matches. If it does not match, Linux shell regards that the user input is unfinished and waits for more input instead of entering the <code>gsql</code> program.</td>
<td></td>
</tr>
<tr>
<td>-L, --log-file=FILENAME</td>
<td>Writes normal output destination and all query output into the <code>FILENAME</code> file.</td>
<td><strong>CAUTION</strong> An absolute path or relative path that meets the OS path naming convention</td>
</tr>
<tr>
<td></td>
<td><strong>CAUTION</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• When this parameter is used in some SQL statements, sensitive information, such as user passwords, may be disclosed. Use this parameter with caution.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• This parameter retains only the query result in the corresponding file, so that the result can be easily found and parsed by other invokers (for example, automatic O&amp;M scripts). Logs about <code>gsql</code> operation are not retained.</td>
<td></td>
</tr>
<tr>
<td>-m, --maintenance</td>
<td>Allows a cluster to be connected when a two-phase transaction is being restored.</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong> The parameter is for engineers only. When this parameter is used, <code>gsql</code> can be connected to the standby server to check data consistency between the primary server and standby server.</td>
<td></td>
</tr>
<tr>
<td>-n, --no-libedit</td>
<td>Closes the command line editing.</td>
<td>-</td>
</tr>
<tr>
<td>-o, --output=FILENAME</td>
<td>Puts all query output into the <code>FILENAME</code> file.</td>
<td><strong>An absolute path or relative path that meets the OS path naming convention</strong></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>-q, --quiet</td>
<td>Indicates the quiet mode and no additional information will be printed.</td>
<td>By default, <code>gsql</code> displays various information.</td>
</tr>
<tr>
<td>-s, --single-step</td>
<td>Runs in single-step mode. This indicates that the user is prompted before each command is sent to the server. This parameter can also be used for canceling execution. This parameter can be used to debug scripts. <strong>CAUTION</strong> When this parameter is used in some SQL statements, sensitive information, such as user passwords, may be disclosed. Use this parameter with caution.</td>
<td>-</td>
</tr>
<tr>
<td>-S, --single-line</td>
<td>Runs in single-row mode where a new line terminates an SQL statement in the same manner as a semicolon does.</td>
<td>-</td>
</tr>
</tbody>
</table>

### Table 3-9 Parameters specifying output formats

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>-A, --no-align</td>
<td>Switches to unaligned output mode.</td>
<td>The default output mode is aligned.</td>
</tr>
<tr>
<td>-F, --field-separator=STRING</td>
<td>Specifies the field separator. The default is the vertical bar (</td>
<td>).</td>
</tr>
<tr>
<td>-H, --html</td>
<td>Turns on the HTML tabular output.</td>
<td>-</td>
</tr>
<tr>
<td>-P, --pset=VAR[=ARG]</td>
<td>Specifies the print option in the \pset format in the command line. <strong>NOTE</strong></td>
<td>The equal sign (=), instead of the space, is used here to separate the name and value. For example, enter <code>-P format=latex</code> to set the output format to LaTeX.</td>
</tr>
<tr>
<td>-R, --record-separator=STRING</td>
<td>Specifies the record separators.</td>
<td>-</td>
</tr>
<tr>
<td>-r</td>
<td>Enables the function of recording historical operations on the client.</td>
<td>This function is disabled by default.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>-t, --tuples-only</td>
<td>Prints only tuples.</td>
<td>-</td>
</tr>
<tr>
<td>-T, --table-attr=TEXT</td>
<td>Specifies options to be placed within the HTML table tag. Use this parameter with the -H, --html parameter to specify the output to the HTML format.</td>
<td>-</td>
</tr>
<tr>
<td>-x, --expanded</td>
<td>Turns on the expanded table formatting mode.</td>
<td>-</td>
</tr>
<tr>
<td>-z, --field-separator-zero</td>
<td>Sets the field separator in the unaligned output mode to be blank. Use this parameter with the -A, --no-align parameter to switch to unaligned output mode.</td>
<td>-</td>
</tr>
<tr>
<td>-0, --record-separator-zero</td>
<td>Sets the record separator in the unaligned output mode to be blank. Use this parameter with the -A, --no-align parameter to switch to unaligned output mode.</td>
<td>-</td>
</tr>
<tr>
<td>-g</td>
<td>Displays separators for all SQL statements and specified files. <strong>NOTE</strong> The -g parameter must be set with the -f parameter.</td>
<td>-</td>
</tr>
</tbody>
</table>

**Table 3-10** Connection parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h, --host=HOSTNAME</td>
<td>Specifies the host name of the machine on which the server is running or the directory for the Unix-domain socket.</td>
<td>If the host name is omitted, <code>gsql</code> connects to the server of the local host over the Unix domain socket or over TCP/IP to connect to local host without the Unix domain socket. The default value is <strong>8000</strong>.</td>
</tr>
<tr>
<td>-p, --port=PORT</td>
<td>Specifies the port number of the database server. You can modify the default port number using the -p, --port=PORT parameter.</td>
<td>The default value is <strong>8000</strong>.</td>
</tr>
</tbody>
</table>
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>-U, --username=USER NAME</td>
<td>Specifies the user that accesses a database.</td>
<td>A character string. The default user is the current user that operates the system.</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• If a user is specified to access a database using this parameter, a user password must be provided together for identity verification. You can enter the password interactively or use the -W parameter to specify a password.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• To connect to a database, add an escape character before any dollar sign ($) in the user name.</td>
<td></td>
</tr>
<tr>
<td>-W, --password=PASS WORD</td>
<td>Specifies a password when the -U parameter is used to connect to a remote database.</td>
<td>The password must meet complexity requirements.</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>To connect to a database, add an escape character before any backslash () or back quote (') in the password.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If this parameter is not specified but database connection requires your password, you will be prompted to enter your password in interactive mode. The maximum length of the password is 999 bytes, which is restricted by the value of the GUC parameter password max length.</td>
<td></td>
</tr>
</tbody>
</table>

### 3.5 Meta-Command Reference

This section describes meta-commands provided by gsql after the DWS database CLI tool is used to connect to a database. A gsql meta-command can be anything that you enter in gsql and begins with an unquoted backslash.

**Precautions**

- The format of the gsql meta-command is a backslash (\) followed by a command verb, and then a parameter. The parameters are separated from the command verb and from each other by any number of whitespace characters.
- To include whitespace in a parameter, you can quote it with single quotation marks (\'). To include single quotation marks in a parameter, add a backslash in front of it. Anything contained in single quotation marks is furthermore subject to C-like substitutions for \n (new line), \t (tab), \b (backspace), \r (carriage return), \f (form feed), \d (octal), and \x (hexadecimal).
- Within a parameter, text enclosed in double quotation marks (""") is taken as a command line input to the shell. The command output (with any trailing newline removed) is taken as a parameter.
- If an unquoted argument begins with a colon (:), the parameter is taken as a gsql variable and the value of the variable is used as the parameter value instead.
Some commands take an SQL identifier (such as a table name) as a parameter. These parameters follow the SQL syntax rules: Unquoted letters are forced to lowercase, while double quotation marks (""") protect letters from case conversion and allow incorporation of whitespace into the identifier. Within double quotation marks, paired double quotation marks reduce to a single double quotation mark in the result name. For example, FOO"BAR"BAZ is interpreted as fooBARbaz, and "A weird""name" becomes A weird"name.

Parameter parsing stops when another unquoted backslash appears. An unquoted backslash is taken as the beginning of a new meta-command. The special sequence \ (two backslashes) marks the end of parameters and continues parsing SQL statements if any. In this way, SQL statements and gsql commands can be freely mixed in a line. However, the parameters of a meta-command cannot continue beyond the end of a line in any situations.

Meta-command

For details about meta-commands, see Table 3-11, Table 3-12, Table 3-13, Table 3-14, Table 3-16, Table 3-18, Table 3-19, Table 3-20, and Table 3-22.

**NOTICE**

FILE mentioned in the following commands indicates a file path. This path can be an absolute path such as /home/ gauss/file.txt or a relative path, such as file.txt. By default, a file.txt is created in the path where the user runs gsql commands.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\copyright</td>
<td>Displays DWS version and copyright information.</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>\g [FILE] or ;</td>
<td>Performs a query operation and sends the result to a file or pipe.</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>\h(\help) [NAME]</td>
<td>Provides syntax help on the specified SQL statement.</td>
<td>If the name is not specified, then gsql will list all the commands for which syntax help is available. If the name is an asterisk (*), syntax help on all SQL statements is displayed.</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
<td></td>
</tr>
</tbody>
</table>
| `\parallel [on [num]] off` | Controls the parallel execution function.  
  - **on**: The switch is enabled and the maximum number of concurrently executed tasks is `num`.  
  - **off**: This switch is disabled.  
  **NOTE**  
  - Parallel execution is not allowed in a running transaction and a transaction is not allowed to be stared during parallel execution.  
  - Parallel execution of `\d` meta-commands is not allowed.  
  - If `SELECT` statements are run concurrently, customers can accept the problem that the return results are displayed randomly but they cannot accept it if a core dump or process response failure occurs.  
  - `SET` statements are not allowed in concurrent tasks because they may cause unexpected results.  
  - Temporary tables cannot be created. If temporary tables are required, create them before parallel execution is enabled, and use them only in the parallel execution. Temporary tables cannot be created in parallel execution.  
  - When `\parallel` is executed, `num` independent gsql processes can be connected to the database server.  
  - The duration of all the jobs specified using `\parallel` cannot exceed `session_timeout` in `Data Warehouse Service Database Developer Guide`. Otherwise, the database may be disconnected during concurrent execution.  
  The default value of `num` is **1024**.  
  **NOTICE**  
  - The maximum number of connections allowed by the server is determined based on `max_connection` and the number of current connections.  
  - Set the value of `num` based on the allowed number of connections.  | - |

| `\q` | Exits the gsql program. In a script file, this command is run only when a script terminates. | - |

**Table 3-12 Buffer query meta-commands**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\e [FILE] [LINE]</code></td>
<td>Use an external editor to edit the query buffer or file.</td>
</tr>
<tr>
<td><code>\ef [FUNCNAME [LINE]]</code></td>
<td>Use an external editor to edit the function definition. If <code>LINE</code> is specified, the cursor will point to the specified line of the function body.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>\p</td>
<td>Prints the current query buffer to the standard output.</td>
</tr>
<tr>
<td>\r</td>
<td>Resets (clears) the query buffer.</td>
</tr>
<tr>
<td>\w FILE</td>
<td>Outputs the current query buffer to a file.</td>
</tr>
</tbody>
</table>

### Table 3-13 Input and output meta-commands

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\copy { table [ ( column_list ) ]</td>
<td>( query ) } { from</td>
</tr>
<tr>
<td>\echo [STRING]</td>
<td>Writes a character string to the standard output.</td>
</tr>
<tr>
<td>\i FILE</td>
<td>Reads content from <code>FILE</code> and uses them as the input for a query.</td>
</tr>
<tr>
<td>\i+ FILE KEY</td>
<td>Runs commands in an encrypted file.</td>
</tr>
<tr>
<td>\ir FILE</td>
<td>Is similar to <code>\i</code>, but resolves relative path names differently.</td>
</tr>
<tr>
<td>\ir+ FILE KEY</td>
<td>Is similar to <code>\i</code>, but resolves relative path names differently.</td>
</tr>
<tr>
<td>\o [FILE]</td>
<td>Saves all query results to a file.</td>
</tr>
<tr>
<td>\qecho [STRING]</td>
<td>Prints a character string to the query result output.</td>
</tr>
</tbody>
</table>

**NOTE**

In Table 3-14, S indicates that the system object is displayed, and + indicates that additional object descriptions are displayed. PATTERN specifies the name of an object to be displayed.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>\d[S+]</td>
<td>Lists all tables, views, and sequences of all schemas in the search_path. When objects with the same name exist in different schemas in the search_path, only the object in the schema that ranks first in the search_path is displayed.</td>
<td>-</td>
<td>Lists all tables, views, and sequences of all schemas in the search_path. \d</td>
</tr>
<tr>
<td>\d[S+] NAME</td>
<td>Lists the structure of specified tables, views, and indexes.</td>
<td>-</td>
<td>Lists the structure of table a. \dtable a</td>
</tr>
<tr>
<td>\d+ [PATTERN]</td>
<td>Lists all tables, views, and indexes.</td>
<td>If PATTERN is specified, only tables, views, and indexes whose names match PATTERN are displayed.</td>
<td>Lists all tables, views, and indexes whose names start with f. \d+ f*</td>
</tr>
<tr>
<td>\da[S] [PATTERN]</td>
<td>Lists all available aggregate functions, together with their return value types and the data types.</td>
<td>If PATTERN is specified, only aggregate functions whose names match PATTERN are displayed.</td>
<td>Lists all available aggregate functions whose names start with f, together with their return value types and the data types. \da f*</td>
</tr>
<tr>
<td>\db[+] [PATTERN]</td>
<td>Lists all available tablespaces.</td>
<td>If PATTERN is specified, only tablespaces whose names match PATTERN are displayed.</td>
<td>Lists all available tablespaces whose names start with p. \db p*</td>
</tr>
<tr>
<td>\dc[S+] [PATTERN]</td>
<td>Lists all available conversions between character sets.</td>
<td>If PATTERN is specified, only conversions whose names match PATTERN are displayed.</td>
<td>Lists all available conversions between character sets. \dc *</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Example</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>\dC[+]</code></td>
<td>Lists all type conversions.</td>
<td>If <code>PATTERN</code> is specified, only conversions whose names match <code>PATTERN</code> are displayed.</td>
<td>Lists all type conversion whose pattern names start with <code>c</code>. <code>\dC c</code></td>
</tr>
<tr>
<td><code>[PATTER N]</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>\dd[S]</code></td>
<td>Lists descriptions about objects matching <code>PATTERN</code>.</td>
<td>If <code>PATTERN</code> is not specified, all visible objects are displayed. The objects include aggregations, functions, operators, types, relations (table, view, index, sequence, and large object), and rules.</td>
<td>Lists all visible objects. <code>\dd</code></td>
</tr>
<tr>
<td><code>[PATTER N]</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>\ddp</code></td>
<td>Lists all default permissions.</td>
<td>If <code>PATTERN</code> is specified, only permissions whose names match <code>PATTERN</code> are displayed.</td>
<td>Lists all default permissions. <code>\ddp</code></td>
</tr>
<tr>
<td><code>[PATTER N]</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>\dD[S+]</code></td>
<td>Lists all available domains.</td>
<td>If <code>PATTERN</code> is specified, only domains whose names match <code>PATTERN</code> are displayed.</td>
<td>Lists all available domains. <code>\dD</code></td>
</tr>
<tr>
<td><code>[PATTER N]</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>\ded[+]</code></td>
<td>Lists all Data Source objects.</td>
<td>If <code>PATTERN</code> is specified, only objects whose names match <code>PATTERN</code> are displayed.</td>
<td>Lists all Data Source objects. <code>\ded</code></td>
</tr>
<tr>
<td><code>[PATTER N]</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>\det[+]</code></td>
<td>Lists all external tables.</td>
<td>If <code>PATTERN</code> is specified, only tables whose names match <code>PATTERN</code> are displayed.</td>
<td>Lists all external tables. <code>\det</code></td>
</tr>
<tr>
<td><code>[PATTER N]</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Example</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
</tbody>
</table>
| \des[+] [PATTERN] | Lists all external servers. | If PATTERN is specified, only servers whose names match PATTERN are displayed. | Lists all external servers. \
| \deu[+] [PATTERN] | Lists user mappings. | If PATTERN is specified, only information whose name matches PATTERN is displayed. | Lists user mappings. \
| \dew[+] [PATTERN] | Lists foreign-data wrappers. | If PATTERN is specified, only data whose name matches PATTERN is displayed. | Lists foreign-data wrappers. \
| \df[antw][S+] [PATTERN] | Lists all available functions, together with their parameters and return types. a indicates an aggregate function, n indicates a common function, t indicates a trigger, and w indicates a window function. | If PATTERN is specified, only functions whose names match PATTERN are displayed. | Lists all available functions, together with their parameters and return types. \
| \dF[+] [PATTERN] | Lists all text search configurations. | If PATTERN is specified, only configurations whose names match PATTERN are displayed. | Lists all text search configurations. \
<p>| \dFd[+] [PATTERN] | Lists all text search dictionaries. | If PATTERN is specified, only dictionaries whose names match PATTERN are displayed. | Lists all text search dictionaries. |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>\dFp[+] [PATTERN]</td>
<td>Lists all text search parsers.</td>
<td>If PATTERN is specified, only analyzers whose names match PATTERN are displayed.</td>
<td>Lists all text search parsers.</td>
</tr>
<tr>
<td>\dFt[+] [PATTERN]</td>
<td>Lists all text search templates.</td>
<td>If PATTERN is specified, only templates whose names match PATTERN are displayed.</td>
<td>Lists all text search templates.</td>
</tr>
<tr>
<td>\dg[+] [PATTERN]</td>
<td>Lists all database roles.</td>
<td>If PATTERN is specified, only roles whose names match PATTERN are displayed.</td>
<td>List all database roles whose names start with j and end with e.</td>
</tr>
<tr>
<td>\dl</td>
<td>This is an alias for \lo_list, which shows a list of large objects.</td>
<td>-</td>
<td>Lists all large objects.</td>
</tr>
<tr>
<td>\dL[S+] [PATTERN]</td>
<td>Lists available procedural languages.</td>
<td>If PATTERN is specified, only languages whose names match PATTERN are displayed.</td>
<td>Lists available procedural languages.</td>
</tr>
<tr>
<td>\dn[S+] [PATTERN]</td>
<td>Lists all schemas (namespace).</td>
<td>If PATTERN is specified, only schemas whose names match PATTERN are displayed.</td>
<td>Lists information about all schemas whose names start with d.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Example</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>\do{S} [PATTERN]</td>
<td>Lists available operators with their operand and return types.</td>
<td>If PATTERN is specified, only operators whose names match PATTERN are displayed. By default, only operators you created are displayed.</td>
<td>Lists available operators with their operand and return types.</td>
</tr>
<tr>
<td>\dO{S+} [PATTERN]</td>
<td>Lists collations.</td>
<td>If PATTERN is specified, only collations whose names match PATTERN are displayed. By default, only collations you created are displayed.</td>
<td>Lists collations.</td>
</tr>
<tr>
<td>\dp [PATTERN]</td>
<td>Lists tables, views, and related permissions.</td>
<td>If PATTERN is specified, only tables and views whose names match PATTERN are displayed.</td>
<td>Lists tables, views, and related permissions.</td>
</tr>
<tr>
<td>\drds [PATTERN1 [PATTERN2]]</td>
<td>Lists all modified configuration parameters. These settings can be for roles, for databases, or for both. PATTERN1 and PATTERN2 indicate a role pattern and a database pattern, respectively.</td>
<td>If PATTERN is specified, only collations whose names match PATTERN are displayed. If the default value is used or * is specified, all settings are listed.</td>
<td>Lists all modified configuration parameters of the postgres database.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Example</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>\dT[S+] [PATTERN]</td>
<td>Lists all data types.</td>
<td>If PATTERN is specified, only types whose names match PATTERN are displayed.</td>
<td>Lists all data types. \dT</td>
</tr>
<tr>
<td>\du[+] [PATTERN]</td>
<td>Lists all database roles. <strong>NOTE</strong> Since the concepts of &quot;users&quot; and &quot;groups&quot; have been unified into &quot;roles&quot;, this command is now equivalent to \dg. The two commands are all reserved for forward compatibility.</td>
<td>If PATTERN is specified, only roles whose names match PATTERN are displayed.</td>
<td>Lists all database roles. \du</td>
</tr>
<tr>
<td>\dE[S+] [PATTERN] \di[S+] [PATTERN] \ds[S+] [PATTERN] \dt[S+] [PATTERN] \dv[S+] [PATTERN]</td>
<td>In this group of commands, the letters E, i, s, t, and v stand for a foreign table, index, sequence, table, or view, respectively. You can specify any or a combination of these letters sequenced in any order to obtain an object list. For example, \dit lists all indexes and tables. If a command is suffixed with a plus sign (+), physical dimensions and related descriptions of each object will be displayed. <strong>NOTE</strong> This version does not support sequences.</td>
<td>If PATTERN is specified, only objects whose names match PATTERN are displayed. By default, only objects you created are displayed. You can specify PATTERN or S to view other system objects.</td>
<td>Lists all indexes and views. \div</td>
</tr>
<tr>
<td>\dx[+] [PATTERN]</td>
<td>Lists installed extensions.</td>
<td>If PATTERN is specified, only extensions whose names match PATTERN are displayed.</td>
<td>Lists installed extensions. \dx</td>
</tr>
<tr>
<td>\l[+]</td>
<td>Lists the names, owners, character set encoding, and permissions of all databases on the server.</td>
<td>-</td>
<td>Lists the names, owners, character set encoding, and permissions of all databases on the server. \l</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Example</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>$\text{FUNCNAME}$</td>
<td>Shows function definitions. <strong>NOTE</strong> If the function name contains parentheses, enclose the function name with quotation marks and add the parameter type list following the double quotation marks. Also enclose the list with parentheses.</td>
<td>-</td>
<td>Assume a function <code>function_a</code> and a function <code>func()</code>name. This parameter will be as follows: \sf function_a \sf &quot;func()name&quot;(argtype1, argtype2)</td>
</tr>
<tr>
<td>$\text{PATTERN}$</td>
<td>Lists all tables, views, and sequences in the database and their access permissions.</td>
<td>If a pattern is given, it is a regular expression, and only matched tables, views, and sequences are displayed.</td>
<td>Lists all tables, views, and sequences in the database and their access permissions. $\text{PATTERN}$</td>
</tr>
</tbody>
</table>

Table 3-15 Permission descriptions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>SELECT: allows users to read data from specified tables and views.</td>
</tr>
<tr>
<td>w</td>
<td>UPDATE: allows users to update columns for specified tables.</td>
</tr>
<tr>
<td>a</td>
<td>INSERT: allows users to insert data to specified tables.</td>
</tr>
<tr>
<td>d</td>
<td>DELETE: allows users to delete data from specified tables.</td>
</tr>
<tr>
<td>D</td>
<td>TRUNCATE: allows users to delete all data from specified tables.</td>
</tr>
<tr>
<td>x</td>
<td>REFERENCES: allows users to create foreign key constraints.</td>
</tr>
<tr>
<td>t</td>
<td>TRIGGER: allows users to create a trigger on specified tables.</td>
</tr>
<tr>
<td>X</td>
<td>EXECUTE: allows users to use specified functions and the operators that are realized by the functions.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Usage:</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
</tr>
<tr>
<td>U</td>
<td>For procedural languages, allows users to specify a procedural language when creating a function.</td>
</tr>
<tr>
<td></td>
<td>For schemas, allows users to access objects included in specified schemas.</td>
</tr>
<tr>
<td></td>
<td>For sequences, allows users to use the nextval function.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>Create:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For databases, allows users to create schemas within a database.</td>
</tr>
<tr>
<td></td>
<td>For schemas, allows users to create objects in a schema.</td>
</tr>
<tr>
<td></td>
<td>For tablespaces, allows users to create tables in a tablespace and set the tablespace to default one when creating databases and schemas.</td>
</tr>
</tbody>
</table>

| c         | Connect: allows users to access specified databases. |
| T         | Temporary: allows users to create temporary tables. |
| arwdDxt   | All Privileges: grants all available permissions to specified users or roles at a time. |
| *         | Authorization options for preceding permissions |

---

**Table 3-16 Formatting meta-commands**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\a</td>
<td>Controls the switchover between unaligned mode and aligned mode.</td>
</tr>
<tr>
<td>\C [STRING]</td>
<td>Sets the title of any table being printed as the result of a query or cancels such a setting.</td>
</tr>
<tr>
<td>\f [STRING]</td>
<td>Sets a field separator for unaligned query output.</td>
</tr>
<tr>
<td>\H</td>
<td>• If the text format schema is used, switches to the HTML format.</td>
</tr>
<tr>
<td></td>
<td>• If the HTML format schema is used, switches to the text format.</td>
</tr>
<tr>
<td>\pset NAME [VALUE]</td>
<td>Sets options affecting the output of query result tables. For details about the value of NAME, see Table 3-17.</td>
</tr>
<tr>
<td>\t [on</td>
<td>off]</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>\T [STRING]</td>
<td>Specifies attributes to be placed within the table tag in HTML output format. If the parameter is not configured, the attributes are not set.</td>
</tr>
<tr>
<td>\x [on</td>
<td>off</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
</table>
| border   | The value must be a number. In general, a larger number indicates wider borders and more table lines. | • The value is an integer greater than 0 in HTML format.  
  • The value range in other formats is as follows:  
    - 0: no border  
    - 1: internal dividing line  
    - 2: table frame |
| expanded (or x) | Switches between regular and expanded formats. | • When expanded format is enabled, query results are displayed in two columns, with the column name on the left and the data on the right. This format is useful if the data does not fit the screen in the normal "horizontal" format.  
  • The expanded format is used when the query output is wider than the screen. Otherwise, the regular format is used. The regular format is effective only in the aligned and wrapped formats. |
<p>| fieldsep | Specifies the field separator to be used in unaligned output format. In this way, you can create tab- or comma-separated output required by other programs. To set a tab as field separator, type \set fieldsep \t. The default field separator is a vertical bar ('|'). | - |</p>
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldsep_zero</td>
<td>Sets the field separator to be used in unaligned output format to zero bytes.</td>
<td>-</td>
</tr>
<tr>
<td>footer</td>
<td>Enables or disables the display of table footers.</td>
<td>-</td>
</tr>
</tbody>
</table>
| format      | Selects the output format. Unique abbreviations are allowed. (That means a single letter is sufficient.) | Value range:  
  - unaligned: Write all columns of a row on one line, separated by the currently active column separator.  
  - aligned: This format is standard and human-readable.  
  - wrapped: This format is similar to aligned, but includes the packaging cross-line width data value to suit the width of the target field output.  
  - html: This format output table to the markup language for a document. The output is not a complete document.  
  - latex: This format output table to the markup language for a document. The output is not a complete document.  
  - troff-ms: This format output table to the markup language for a document. The output is not a complete document. |
| null        | Sets a character string to be printed in place of a null value.             | By default, nothing is printed, which can easily be mistaken for an empty character string. |
| numericlocale | Enables or disables the display of a locale-specific character to separate groups of digits to the left of the decimal marker. | - on: The specified separator is displayed.  
  - off: The specified separator is not displayed  
  If this parameter is ignored, the default separator is displayed. |
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>pager</td>
<td>Controls the use of a pager for query and gsql help outputs. If the PAGER environment variable is set, the output is piped to the specified program. Otherwise, a platform-dependent default is used.</td>
<td>• <strong>on</strong>: The pager is used for terminal output that does not fit the screen.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>off</strong>: The pager is not used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>always</strong>: The pager is used for all terminal output regardless of whether it fits the screen.</td>
</tr>
<tr>
<td>recordsep</td>
<td>Specifies the record separator to be used in unaligned output format.</td>
<td>-</td>
</tr>
<tr>
<td>recordsep_zero</td>
<td>Specifies the record separator to be used in unaligned output format to zero bytes.</td>
<td>-</td>
</tr>
<tr>
<td>tableattr (or T)</td>
<td>Specifies attributes to be placed inside the HTML table tag in HTML output format (such as cellpadding or bgcolor). Note that you do not need to specify border here because it has been used by \set border. If no value is given, the table attributes do not need to be set.</td>
<td>-</td>
</tr>
<tr>
<td>title</td>
<td>Specifies the table title for any subsequently printed tables. This can be used to give your output descriptive tags. If no value is given, the title does not need to be set.</td>
<td>-</td>
</tr>
<tr>
<td>tuples_only (or t)</td>
<td>Enables or disables the tuples-only mode. Full display may show extra information, such as column headers, titles, and footers. In tuples-only mode, only the table data is displayed.</td>
<td>-</td>
</tr>
</tbody>
</table>
### Table 3-18 Connection meta-commands

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[connect] [DBNAME]- [USER]- [HOST]- [PORT]-</td>
<td>Connects to a new database. (The current database is <strong>postgres</strong>.) If a database name contains more than 63 bytes, only the first 63 bytes are valid and are used for connection. However, the database name displayed in the command line of <strong>gsql</strong> is still the name before the truncation. <strong>NOTE</strong> If the database login user is changed during reconnection, you need to enter the password of the new user. The maximum length of the password is 999 bytes, which is restricted by the value of the GUC parameter password max length.</td>
<td>-</td>
</tr>
<tr>
<td>[encoding] [ENCODING]</td>
<td>Sets the client character set encoding. This command shows the current encoding if it has no parameter.</td>
<td></td>
</tr>
<tr>
<td>[conninfo]</td>
<td>Outputs information about the current database connection.</td>
<td>-</td>
</tr>
</tbody>
</table>

### Table 3-19 OS meta-commands

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[cd] [DIR]</td>
<td>Changes the current working directory.</td>
<td>An absolute path or relative path that meets the OS path naming convention</td>
</tr>
<tr>
<td>[setenv] NAME [VALUE]</td>
<td>Sets the <strong>NAME</strong> environment variable to <strong>VALUE</strong>. If <strong>VALUE</strong> is not provided, do not set the environment variable.</td>
<td>-</td>
</tr>
<tr>
<td>[timing] [on</td>
<td>off]</td>
<td>Toggles a display of how long each SQL statement takes, in milliseconds.</td>
</tr>
<tr>
<td>! [COMMAND]</td>
<td>Escapes to a separate Unix shell or runs a Unix command.</td>
<td>-</td>
</tr>
</tbody>
</table>
### Table 3-20 Variable meta-commands

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\prompt [TEXT] NAME</td>
<td>Prompts the user to use texts to specify a variable name.</td>
</tr>
<tr>
<td>\set [NAME [VALUE]]</td>
<td>Sets the NAME internal variable to VALUE. If more than one value is provided, NAME is set to the concatenation of all of them. If only one parameter is provided, the variable is set with an empty value. Some common variables are processed in another way in gsql, and they are the combination of uppercase letters, numbers, and underscores. Table 3-21 describes a list of variables that are processed in a way different from other variables.</td>
</tr>
<tr>
<td>\unset NAME</td>
<td>Deletes the variable name of gsql.</td>
</tr>
</tbody>
</table>

### Table 3-21 Common \set commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>\set VERBOSITY value</td>
<td>This variable can be set to default, verbose, or terse to control redundant lines of error reports.</td>
<td>Value range: default, verbose, terse</td>
</tr>
<tr>
<td>\set ON_ERROR_STOP P value</td>
<td>If this variable is set, the script execution stops immediately. If this script is invoked from another script, that script will be stopped immediately as well. If the primary script is invoked using the -f option rather than from one gsql session, gsql will return error code 3, indicating the difference between the current error and critical errors. (The error code for critical errors is 1.)</td>
<td>Value range: on/off, true/false, yes/no, 1/0</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Value Range</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td><code>\set RETRY</code> <img src="toggle" alt="retry_times" /></td>
<td>Determines whether to enable the retry function if statement execution encounters errors. The parameter <code>retry_times</code> specifies the maximum number of retry times and the default value is 5. Its value ranges from 5 to 10. If the retry function has been enabled, when you run the <code>\set RETRY</code> command again, the retry function will be disabled.</td>
<td>Value range of <code>retry_times</code>: 5 to 10</td>
</tr>
<tr>
<td></td>
<td>The configuration file <code>retry_errcodes.conf</code> shows a list of errors. If these errors occur, retry is required. This configuration file is placed in the same directory as that for executable <code>gsql</code> programs. This configuration file is configured by the system rather than by users and cannot be modified by the users.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The retry function can be used in the following 13 error scenarios:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• YY001: TCP communication errors. Print information: <code>Connection reset by peer</code>. (reset between CN and DN)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• YY002: TCP communication errors. Print information: <code>Connection reset by peer</code>. (reset between DN and DN)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• YY003: Lock timeout. Print information: <code>Lock wait timeout.../wait transaction xxx sync time exceed xxx</code>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• YY004: TCP communication errors. Print information: <code>Connection timed out</code>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• YY005: Failed to issue SET commands. Print information: <code>ERROR SET query</code>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• YY006: Failed to apply for memory. Print information: <code>memory is temporarily unavailable</code>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• YY007: Communication library error. Print information: <code>Memory allocate error</code>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• YY008: Communication library error. Print information: <code>No data in buffer</code>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• YY009: Communication library error. Print information: <code>Close because release memory</code>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• YY010: Communication library error. Print information: <code>TCP disconnect</code>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• YY011: Communication library error. Print information: <code>SCTP disconnect</code>.</td>
<td></td>
</tr>
</tbody>
</table>
### Command

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>YY012: Communication library error. Print information: <strong>Stream closed by remote.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YY013: Communication library error. Print information: <strong>Wait poll unknown error.</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If an error occurs, `gsql` queries connection status of all CNs and DN. If the connection status is abnormal, `gsql` sleeps for 1 minute and tries again. In this case, the retries in most of the primary/standby switchover scenarios are involved.

**NOTE**

1. Statements in transaction blocks cannot be retried upon a failure.
2. Retry is not supported if errors are found using ODBC or JDBC.
3. For SQL statements with unlogged tables, the retry is not supported if a node is faulty.
4. If a CN or GTM is faulty, the retry on the `gsql` client is not supported.
5. For `gsql` client faults, the retry is not supported.

### Table 3-22 Large object meta-commands

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\lo_list</td>
<td>Shows a list of all DWS large objects stored in the database, along with the comments provided for them.</td>
</tr>
</tbody>
</table>

### PATTERN

The various `\d` commands accept a **PATTERN** parameter to specify the object name to be displayed. In the simplest case, a pattern is just the exact name of the object. The characters within a pattern are normally folded to lower case, similar to those in SQL names. For example, `\dt FOO` will display the table named `foo`. As in SQL names, placing double quotation marks (""") around a pattern prevents them being folded to lower case. If you need to include a double quotation mark ("") in a pattern, write it as a pair of double quotation marks (""") within a double-quote sequence, which is in accordance with the rules for SQL quoted identifiers. For example, `\dt "FOO"BAR` will be displayed as a table named `FOO"BAR` instead of `foo"bar`. You cannot put double quotation marks around just part of a pattern, which is different from the normal rules for SQL names. For example, `\dt FOO"FOO"BAR` will be displayed as a table named `fooFOObar` if just part of a pattern is quoted.

Whenever the **PATTERN** parameter is omitted completely, the `\d` commands display all objects that are visible in the current schema search path, which is
equivalent to using an asterisk (*) as the pattern. An object is regarded to be visible if it can be referenced by name without explicit schema qualification. To see all objects in the database regardless of their visibility, use a dot within double quotation marks (\*\*) as the pattern.

Within a pattern, the asterisk (*) matches any sequence of characters (including no characters) and a question mark (?) matches any single character. This notation is comparable to Unix shell file name patterns. For example, `\dt int*` displays tables whose names begin with `int`. But within double quotation marks, the asterisk (*) and the question mark (?) lose these special meanings and are just matched literally.

A pattern that contains a dot (\.) is interpreted as a schema name pattern followed by an object name pattern. For example, `\dt foo.*,bar*` displays all tables (whose names include `bar`) in schemas starting with `foo`. If no dot appears, then the pattern matches only visible objects in the current schema search path. Again, a dot within double quotation marks loses its special meaning and is matched literally.

Advanced users can use regular-expression notations, such as character classes. For example, `[0-9]` can be used to match any digit. All special characters in regular expressions, except the following, work as specified in POSIX regular expressions in \textit{Data Warehouse Service Database Development Guide}:

- A dot (\.) is used as a separator.
- An asterisk (*) is translated into an asterisk prefixed with a dot (\.*), which is a regular-expression marking.
- A question mark (?) is translated into a dot (\.).
- A dollar sign ($) is matched literally.

You can write `\?`, `(R+|)`, `(R|)`, and `R` to the following pattern characters: \., `R*`, and `R?`. The dollar sign ($) does not need to work as a regular-expression character since the pattern must match the whole name, which is different from the usual interpretation of regular expressions. In other words, the dollar sign ($) is automatically appended to your pattern. If you do not expect a pattern to be anchored, write an asterisk (*) at its beginning or end. All regular-expression special characters within double quotation marks lose their special meanings and are matched literally. Regular-expression special characters in operator name patterns (such as the `\do` parameter) are also matched literally.

### 3.6 Troubleshooting

**Low Connection Performance**

- **log_hostname** is enabled, but DNS is incorrect.
  
  Connect to the database, run `show log_hostname` to check whether `log_hostname` is enabled in the database.
  
  If it is enabled, the database kernel will use DNS to check the name of the host where the client is deployed. If the host where the database CN resides is configured with an incorrect or unreachable DNS server, the database connection will take a long time to set up. For more information about this parameter, see the description about `log_hostname` in the section about log content in \textit{Data Warehouse Service Database Developer Guide}. 


The database kernel slowly runs the initialization statement. Problems are difficult to locate in this scenario. Try using the `strace` Linux trace command.

```
strace gsql -U MyUserName -W MyPassWord -d postgres -h 127.0.0.1 -p 23508 -r -c \"q\"
```

The database connection process will be printed on the screen. If the following statement takes a long time to run:

```
sendto(3, "Q\0\0\0\25SELECT VERSION()\0", 22, MSG_NOSIGNAL, NULL, 0) = 22
poll([{{fd=3, events=POLLIN|POLLERR}}, 1]. -1) = 1 ({{fd=3, revents=POLLIN}})
```

It indicates that `SELECT VERSION()` statement was run slowly.

After the database is connected, you can run the `explain performance select version()` statement to find the reason why the initialization statement was run slowly. For more information, see the section about SQL execution plans in *Data Warehouse Service Database Developer Guide*.

An uncommon scenario is that the disk of the machine where the CN resides is full or faulty, affecting queries and leading to user authentication failures. As a result, the connection process is suspended. To solve this problem, simply clear the data disk space of the CN.

TCP connection is set up slowly.

Adapt the steps of troubleshooting slow initialization statement execution. Use `strace`. If the following statement was run slowly:

```
connect(3, {sa_family=AF_FILE, path="/home/test/tmp/gaussdb_llt1/.s.PGSQL.61052"}, 110) = 0
```

Or

```
connect(3, {sa_family=AF_INET, sin_port=htons(61052), sin_addr=inet_addr("127.0.0.1")}, 16) = -1
EINPROGRESS (Operation now in progress)
```

It indicates that the physical connection between the client and the database was set up slowly. In this case, check whether the network is unstable or has high throughput.

### Problems in Setting Up Connections

- **gsql: could not connect to server: No route to host**
  This problem occurs generally because an unreachable IP address or port number was specified. Check whether the values of `-h` and `-p` parameters are correct.

- **gsql: FATAL: Invalid username/password, login denied.**
  This problem occurs generally because an incorrect user name or password was entered. Contact the database administrator to check whether the user name and password are correct.

- The "libpq.so" loaded mismatch the version of `gsql`, please check it.
  This problem occurs because the version of `libpq.so` used in the environment does not match that of `gsql`. Run the `ldd gsql` command to check the version of the loaded `libpq.so`, and then load correct `libpq.so` by modifying the environment variable `LD_LIBRARY_PATH`.

- **gsql: symbol lookup error: xxx/gsql: undefined symbol: libpqVersionString**
  This problem occurs because the version of `libpq.so` used in the environment does not match that of `gsql` (or the PostgreSQL `libpq.so` exists in the environment). Run the `ldd gsql` command to check the version of the loaded `libpq.so`, and then load correct `libpq.so` by modifying the environment variable `LD_LIBRARY_PATH`.  


gsql: connect to server failed: Connection timed out
Is the server running on host "xx.xxx.xxx.xxx" and accepting TCP/IP connections on port xxxx?
This problem is caused by network connection faults. Check the network connection between the client and the database server. If you cannot ping from the client to the database server, the network connection is abnormal. Contact network management personnel for troubleshooting.
```
ping -c 4 10.10.10.1
PING 10.10.10.1 (10.10.10.1) 56(84) bytes of data.
From 10.10.10.1: icmp_seq=2 Destination Host Unreachable
From 10.10.10.1 icmp_seq=2 Destination Host Unreachable
From 10.10.10.1 icmp_seq=3 Destination Host Unreachable
From 10.10.10.1 icmp_seq=4 Destination Host Unreachable
--- 10.10.10.1 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 2999ms
```
gsql: FATAL: permission denied for database "postgres"
DETAIL: User does not have CONNECT privilege.
This problem occurs because the user does not have the permission to access the database. To solve this problem, perform the following steps:

a. Connect to the database as the system administrator dbadmin.
   ```
gsql -d postgres -U dbadmin -p 8000
```
b. Grant the users with the access permission to the database.
   ```
GRANT CONNECT ON DATABASE postgres TO user1;
```

Common misoperations may also cause a database connection failure, for example, entering an incorrect database name, user name, or password. In this case, the client tool will display the corresponding error messages.
```
gsql -d postgres -p 8000
```
```
gsql: FATAL: database "postgres" does not exist
```
```
gsql -d postgres -U user1 -W gauss@789 -p 8000
```
```
gsql: FATAL: Invalid username/password, login denied.
```
gsql: FATAL: sorry, too many clients already, active/non-active: 197/3.
This problem occurs because the number of system connections exceeds the allowed maximum. Contact the database administrator to release unnecessary sessions.
You can check the number of connections as described in Table 3-23.
You can view the session status in the PG_STAT_ACTIVITY view. To release unnecessary sessions, use the pg_terminate_backend function.
```
select datid,pid,state from pg_stat_activity;
```
```
datid |       pid       | state
-------+-----------------+--------
13205 | 139834762094352 | active
13205 | 139834759993104 | idle
(2 rows)
```
The value of pid is the thread ID of the session. Terminate the session using its thread ID.
```
SELECT PG_TERMINATE_BACKEND(139834759993104);
```
If information similar to the following is displayed, the session is successfully terminated:
```
PG_TERMINATE_BACKEND
----------------------
### Table 3-23 Viewing the numbers of connections

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>View the upper limit of a user's connections.</td>
<td>Run the following command to view the upper limit of user <strong>USER1</strong>'s connections. -1 indicates that no connection upper limit is set for user <strong>USER1</strong>.</td>
</tr>
<tr>
<td></td>
<td><code>SELECT ROLNAME,ROLCONNLIMIT FROM PG_ROLES WHERE ROLNAME='user1';</code></td>
</tr>
<tr>
<td></td>
<td>`rolname</td>
</tr>
<tr>
<td></td>
<td><code>----------+--------------</code></td>
</tr>
<tr>
<td></td>
<td>`user1</td>
</tr>
<tr>
<td></td>
<td>(1 row)</td>
</tr>
<tr>
<td>View the number of connections that have been used by a user.</td>
<td>Run the following command to view the number of connections that have been used by user <strong>user1</strong>. 1 indicates the number of connections that have been used by user <strong>user1</strong>.</td>
</tr>
<tr>
<td></td>
<td><code>SELECT COUNT(*) FROM V$SESSION WHERE USERNAME='user1';</code></td>
</tr>
<tr>
<td></td>
<td><code>count ------</code></td>
</tr>
<tr>
<td></td>
<td><code>1</code></td>
</tr>
<tr>
<td></td>
<td>(1 row)</td>
</tr>
<tr>
<td>View the upper limit of connections to database.</td>
<td>Run the following command to view the upper limit of connections used by <strong>postgres</strong>. -1 indicates that no upper limit is set for the number of connections that have been used by <strong>postgres</strong>.</td>
</tr>
<tr>
<td></td>
<td><code>SELECT DATNAME,DATCONNLIMIT FROM PG_DATABASE WHERE DATNAME='postgres';</code></td>
</tr>
<tr>
<td></td>
<td>`datname</td>
</tr>
<tr>
<td></td>
<td><code>----------+--------------</code></td>
</tr>
<tr>
<td></td>
<td>`postgres</td>
</tr>
<tr>
<td></td>
<td>(1 row)</td>
</tr>
<tr>
<td>View the number of connections that have been used by a database.</td>
<td>Run the following command to view the number of connections that have been used by <strong>postgres</strong>. 1 indicates the number of connections that have been used by <strong>postgres</strong>.</td>
</tr>
<tr>
<td></td>
<td><code>SELECT COUNT(*) FROM PG_STAT_ACTIVITY WHERE DATNAME='postgres';</code></td>
</tr>
<tr>
<td></td>
<td><code>count ------</code></td>
</tr>
<tr>
<td></td>
<td><code>1</code></td>
</tr>
<tr>
<td></td>
<td>(1 row)</td>
</tr>
<tr>
<td>View the total number of connections that have been used by users.</td>
<td>Run the following command to view the number of connections that have been used by users:</td>
</tr>
<tr>
<td></td>
<td><code>SELECT COUNT(*) FROM V$SESSION;</code></td>
</tr>
<tr>
<td></td>
<td><code>count ------</code></td>
</tr>
<tr>
<td></td>
<td><code>10</code></td>
</tr>
<tr>
<td></td>
<td>(1 row)</td>
</tr>
</tbody>
</table>

- `gsql: wait xxx.xxx.xxx.xxx:xxxx timeout expired`
When `gsql` initiates a connection request to the database, a 5-minute timeout period is used. If the database cannot correctly authenticate the client request and client identity within this period, `gsql` will exit the connection process for the current session, and will report the above error.

Generally, this problem is caused by the incorrect host and port (that is, the `xxx` part in the error information) specified by the `-h` and `-p` parameters. As a result, the communication fails. Occasionally, this problem is caused by network faults. To resolve this problem, check whether the host name and port number of the database are correct.

- `gsql`: could not receive data from server: Connection reset by peer.
  Check whether CN logs contain information similar to "FATAL: cipher file "/data/coordinator/server.key.cipher" has group or world access". This error is usually caused by incorrect tampering with the permissions for data directories or some key files. For details about how to correct the permissions, see related permissions for files on other normal instances.

- `gsql`: FATAL: GSS authentication method is not allowed because XXXX user password is not disabled.
  In `pg_hba.conf` of the target CN, the authentication mode is set to `gss` for authenticating the IP address of the current client. However, this authentication algorithm cannot authenticate clients. Change the authentication algorithm to `sha256` and try again.

**NOTE**

- Do not modify the configurations of database cluster hosts in the `pg_hba.conf` file. Otherwise, the database may become faulty.
- You are advised to deploy service applications outside the database cluster.

**Other Faults**

- There is a core dump or abnormal exit due to the bus error.
  Generally, this problem is caused by changes in loading the shared dynamic library (.so file in Linux) during process running. Alternatively, if the process binary file changes, the execution code for the OS to load machines or the entry for loading a dependent library will change accordingly. In this case, the OS kills the process for protection purposes, generating a core dump file.
  To resolve this problem, try again. In addition, do not run service programs in a cluster during O&M operations, such as an upgrade, preventing such a problem caused by file replacement during the upgrade.

**NOTE**

A possible stack of the core dump file contains `dl_main` and its function calling. The file is used by the OS to initialize a process and load the shared dynamic library. If the process has been initialized but the shared dynamic library has not been loaded, the process cannot be considered completely started.
4 Data Studio - Integrated Database Development Tool

4.1 About Data Studio

4.1.1 Overview

Data Studio provides a graphical interface which supports essential features of the database. This simplifies database development and application building tasks.

Data Studio allows the database developer to

- Manage and Create database objects
- Executing SQL statements or SQL scripts
- Editing and executing PL/SQL statements
- Importing and Exporting table data

Creating database objects include

- database
- schema
- functions
- procedures
- tables
- sequences
- columns
- indexes
- constraints
- views
- table spaces
Data Studio also allows the database developer to debug and fix defects in the PL/SQL code using debug operations such as step into, step out, step over, continue, and terminate.

The following figure provides the operational context of database and Data Studio:

![Operational Context Diagram]

### 4.1.2 Constraints and Limitations

The following are known limitations in Data Studio:

**Character Encoding**

When viewing SQL, DDL, object names or data containing Chinese characters, Data Studio encoding needs to be set to GBK provided OS supports GBK. For more information on changing encoding settings, refer to Environment > Session Setting.

**Connection Management**

Comma is considered as delimiter in include/exclude fields in Advanced tab of add and edit connection window. Hence schema name having comma is not supported in include/exclude fields.

**Database Tables**

- In the Create Table wizard > Index tab and in the Create Index wizard, the selected columns between list view, on remove, will not maintain order.
• When an operation has completed, and if the Data Studio window is not the active window of the operating system, then the message dialog is shown only when Data Studio window becomes active.

• The following limitations are applicable for Edit Table Data operations:
  – Entering expression values in Edit Table Data tab is not supported.
  – Data Studio allows editing of only fetched records.
  – Edit table filter feature will not highlight search words within HTML tags such as <, &amp;, >.
  – A cell containing single '&amp;' in it will not be displayed in tooltip. A cell containing two consecutive '&amp;' will display as single '&amp;' in the tooltip.
  – Row focus is not retained on a newly added row. User must click on the desired cell to start editing.

Function/Procedure

Function/Procedure created in SQL Terminal or Create Function/Procedure wizard must end with / to indicate the end of function/procedure. Statements entered after a function/procedure without / at the end will be treated as a single query and may display errors during execution.

General

• A maximum of 100 tabs can be opened in the editor area. Tabs are based on available resources of the host machine.

• A maximum of 64 characters (text only) is allowed for database object names (database, schema, function, procedure, table, sequence, constraint, index, view, and tablespace). There is no limit to the number of characters that can be used in expressions and descriptions in Data Studio.

• A maximum of 300 result tabs can be opened on a logged instance of Data Studio.

• If there are large objects loaded in Object Browser and Search Object window, then expanding of objects in Object Browser may be slow and Data Studio may become unresponsive.

• Resizing the width of a cell containing data exceeding the available display area may cause DS to become unresponsive.

• When the data in a table cell is more than 1000 characters, it will appear trimmed up to 1000 characters with "..." at the end.
  – If the user copies the data from a cell in a table or Result tab and pastes it on any editor (such as SQL terminal/PLSQL source editor, notepad or any other external editor application), the entire data is pasted.
  – If the user copies the data from a cell in a table or Result tab and pastes it on an editable cell (same or different), the cell shows only the first 1000 characters with "..." in the end.
  – When the table/Result tab data is exported, the exported file contains the whole data.
Security

Data Studio validates SSL connection parameters only for the first connection profile. If a second connection is opened, then the connection uses the same SSL connection parameters when the Enable SSL field is checked.

NOTE

For SSL connection, if security files are corrupted, DS will not be able to proceed with any database operation. To recover from this, please remove security folder under the corresponding profile folder and restart DS.

SQL Terminal

- Opening an SQL file containing a large number of queries may result in an 'Insufficient Memory' error. For more information, refer to Troubleshooting.
- Data Studio does not disable the auto-suggest and hyperlink features in commented text in the SQL Terminal.
- Hyperlink feature is not supported if schema or table name have either space or dot (.) in them.
- Auto-suggest is not supported if the object name contains single or double quotes in them.
- DS supports basic formatting of simple SELECT statements only and may not work as expected for complex queries.

4.1.3 Structure of Release Package

The release package structure of Data Studio is as follows:

<table>
<thead>
<tr>
<th>Folders/Files</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>configuration</td>
<td>Contains information about the application launcher and the required Eclipse plug-in path.</td>
</tr>
<tr>
<td>db_assistant</td>
<td>Contains SQL assistant related files.</td>
</tr>
<tr>
<td>Folders/Files</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| **docs**     | - Contains *Data Studio User Manual.pdf* which provides you with details on using Data Studio.  
- Contains copyright notices, licenses, and the written offer for the open source libraries used in Data Studio. |
| **features** | Contains Eclipse (rich client protocol-GUI) and Data Studio features. |
| **p2**       | Contains files required for provisioning and managing Eclipse and Equinox-based applications. |
| **plugins**  | Contains the required Eclipse and Data Studio plugins. |
| **tools**    | Contains Data Studio dependent tools. |
| **UserData/**| Contains separate folders for each OS user who uses Data Studio.  
  - Autosave - Contains the auto saved information of unsaved queries and functions/procedures.  
  - Logs/  
  - Preferences/  
  - Profile/  
    - History/  
  - Security/  
  | NOTE  
  - The UserData folder is created only after the first user opens an instance of Data Studio.  
  - Logs folder, language, memory settings and log level are common for all users.  
  - The Logs folder, *Data Studio.log* file, Preferences folder, *Preferences.prefs* file, Profile folder, connection.properties file, Profiles.txt file, and security folder are created after launching Data Studio.  
  - If Logs folder path is provided in Data Studio.ini file, then logs are created in the specified path.  
  - When user is not able to log in to the Data Studio due to security keys are corrupted.  
  | Follow the steps to generate new security keys:  
  1. Delete the security folder from Data Studio folder -> UserData -> security folder  
  2. Restart Data Studio |
| **artifacts.xml** | Contains the product build information. |
| **changelog.txt** | Contains the detailed change log information of release version. |
| **Data Studio.exe/ DataStudio.sh** | Allows you to connect to the database and perform various operations (like managing database objects, editing or executing PL/SQL programs and so on). |
### 4.1.4 System Requirements

This section provides the minimum system requirements for using Data Studio.

**Software Requirements**

**Operating System Requirements**

The following table lists the details of OS requirements for Data Studio:

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Windows</td>
<td>Windows 2008 (32 bit)</td>
</tr>
<tr>
<td></td>
<td>Windows 2008 (64 bit)</td>
</tr>
<tr>
<td></td>
<td>Windows 7 (32 bit)</td>
</tr>
<tr>
<td></td>
<td>Windows 7 (64 bit)</td>
</tr>
<tr>
<td></td>
<td>Windows 8 (32 bit)</td>
</tr>
<tr>
<td></td>
<td>Windows 8 (64 bit)</td>
</tr>
<tr>
<td></td>
<td>Windows 10 (32 bit)</td>
</tr>
<tr>
<td></td>
<td>Windows 10 (64 bit)</td>
</tr>
</tbody>
</table>

**Browser Requirements**

The following table lists the details of browser requirements for Data Studio:

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Windows</td>
<td>IE 11 and above</td>
</tr>
</tbody>
</table>

**Other Software Requirements**

The following table lists the details of software requirements for Data Studio:
### Table 4-2 List of Software Requirements for Data Studio

<table>
<thead>
<tr>
<th>Software</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>Recommended Open Java Development Kit (JDK) version is 1.8.0_141 or above with appropriate bit number.</td>
</tr>
</tbody>
</table>

### 4.2 Installing and Configuring Data Studio

This section describes the installation and configuration steps to be followed to use Data Studio. It also explains the steps to configure server for debugging PL/SQL Functions.

This section contains the following topics:

- **Installing Data Studio**
- **Configuring Data Studio**
- **Providing Location to Create Log File**
- **Controlling Exception and Error Logs**
- **Description of the Log Message**
- **Different Types of Log Level**

#### Installing Data Studio

Data Studio can be run after extraction of package.

Follow the steps to install Data Studio:

**Step 1** Unzip the required package (32-bit or 64-bit) to the program files (x86) or program files folder respectively. If the user prefer to install in other folder, then admin should control the folder access permissions to users.

You can see the following files and folders:

- Data Studio
- configuration
- db_assistant
- docs
- features
- p2
- plugins
- tools
- UserData
- artifacts.xml
- changelog.txt
- Data Studio.exe
- Data Studio.ini
- readme.txt
Step 2  Locate and double-click **Data Studio.exe** to launch Data Studio.

*NOTE*

*UserData* folder is created after the first user launches Data Studio. Refer to **Getting Started** in case of any error while launching Data Studio.

----End

To create a new database connection, refer to **Adding a Connection**.

**Configuring Data Studio**

Steps to configure Data Studio using **Data Studio.ini** file:

*NOTE*

Restart Data Studio to view parameter changes. Invalid parameters added in the configuration file are ignored by Data Studio. All the following parameters are not mandatory.

List of configuration parameters are used in Data Studio:

<table>
<thead>
<tr>
<th>Table 4-3 Configuration Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>-startup</td>
</tr>
<tr>
<td>--launcher.library</td>
</tr>
<tr>
<td>-clearPersistedState</td>
</tr>
</tbody>
</table>

*NOTE*  It is recommended to add this parameter.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>consoleLineCount</td>
<td>Defines the maximum number of lines to be displayed in the Messages window.</td>
<td>1 - 5000</td>
<td>1000</td>
</tr>
<tr>
<td>-logfolder</td>
<td>Used to create log folder. The user can specify the path to save logs.</td>
<td>N/A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>If the default value &quot;.&quot; is used, then the folder is created in Data Studio\ UserData&lt;username&gt;\logs.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Refer to Providing Location to Create Log File section for more information.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-loginTimeout</td>
<td>Defines the connection open wait time in seconds. Based on the duration value entered Data Studio will try to connect beyond which it throws time out error/connection failed error</td>
<td>N/A</td>
<td>180</td>
</tr>
<tr>
<td>-data</td>
<td>Defines the instance data location for the session.</td>
<td>N/A</td>
<td>@none</td>
</tr>
<tr>
<td>@user.home/MyAppWorkspace</td>
<td>Eclipse workspace is created in this location while Data Studio is being launched. @user.home refers to C:/Users/&lt;username&gt; Eclipse log files are available in @user.home/MyAppWorkspace/.metadata</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
<td>-------------</td>
<td>---------------</td>
</tr>
<tr>
<td>-detailLogging</td>
<td>Defines the criteria with reference to logging error messages. Set to <strong>True</strong> to log all error messages. Set to <strong>False</strong> to log only error messages explicitly mentioned by Data Studio. Refer to <em>Controlling Exception and Error Logs</em> for more information. This parameter is not added by default and it can be set manually if logging is required.</td>
<td>True/False</td>
<td>False</td>
</tr>
<tr>
<td>-logginglevel</td>
<td>Creates the log files based on the value specified. If the value provided is arbitrary or empty, log files will be created according to WARN value. Refer to <em>Different Types of Log Level</em> for more information. This parameter is not added by default and it can be set manually if logging is required.</td>
<td>FATAL, ERROR, WARN, INFO, DEBUG TRACE, ALL, OFF</td>
<td>WARN</td>
</tr>
<tr>
<td>-focusOnFirstResult</td>
<td>Defines auto focus behavior for <strong>Result</strong> window. Set to <strong>false</strong> to automatically set focus to the last opened <strong>Result</strong> window. Set to <strong>true</strong> to disable the automatic set focus.</td>
<td>True/False</td>
<td>False</td>
</tr>
</tbody>
</table>

**NOTE**
- All the above parameters must be added before `-vmargs`.
- `-startup` and `--launcher.library` must be added as first and second parameter respectively.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-vmargs</td>
<td>Specifies the start of virtual machine arguments.</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>NOTE</td>
<td>-vmargs must be the last parameter in the configuration file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-vm &lt;file name (javaw.exe) with relative path to Java executable&gt;</td>
<td>Defines the relative path to Java executable</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>-Dosgi.requiredJavaVersion</td>
<td>Defines the minimum java version required to run Data Studio. This value must not be modified.</td>
<td>N/A</td>
<td>1.8</td>
</tr>
<tr>
<td></td>
<td>Note: Recommended Java version is 1.8.0_141</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Xms</td>
<td>Defines the initial heap space that Data Studio consumes.</td>
<td>N/A</td>
<td>-Xms40m</td>
</tr>
</tbody>
</table>
|                           | This value must be in multiples of 1024 and greater than 40 MB and less than or equal to -Xmx size. Append the letter k or K to indicate kilobytes, m or M to indicate megabytes, g or G to indicate gigabytes. Few examples:  
|                           | ● -Xms40m  
|                           | ● -Xms120m  
<p>|                           | Refer to Java documentation for more information.                           |             |               |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Xmx</td>
<td>Defines the maximum heap space that Data Studio consumes. This value can be</td>
<td>N/A</td>
<td>-Xmx1200m</td>
</tr>
<tr>
<td></td>
<td>modified based on the available RAM space. Append the letter k or K to</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>indicate kilobytes, m or M to indicate megabytes, g or G to indicate</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>gigabytes. Few examples:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>● -Xmx1200m</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>● -Xmx1000m</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Refer to Java documentation for more information.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-OLTPVersionOldST</td>
<td>User can log in to gsql and run the SELECT VERSION() and update the</td>
<td>N/A</td>
<td>-</td>
</tr>
<tr>
<td>-OLTPVersionNewST</td>
<td>User can log in to gsql and run the SELECT VERSION() and update the</td>
<td>N/A</td>
<td>-</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>-------------</td>
<td>---------------</td>
</tr>
</tbody>
</table>
| -testability | This parameter is used to enable testability features. For the current version the features controlled by this parameter with condition as True are:  
- Enables copy content of last triggered auto-complete operation using Ctrl+Space shortcut key.  
- **Execution Plan and Cost with Analyze**  
  displays tree and graphical view of the explain plan. This parameter is not available by default and needs to be added manually for testing. | True/False | False |
| -Duser.language | Defines the language settings for Data Studio. This parameter is added after the language setting is modified. | zh/en | N/A |
| -Duser.country | Defines the country settings for Data Studio. This parameter is added after the language setting is modified. | CN/IN | N/A |

**Providing Location to Create Log File**

**Step 1**  Open the Data Studio.ini file.

**Step 2**  Provide the path for the -logfolder parameter.

For example:
-logfolder=c:\test1
In this case, the Data Studio.log file is created in the c:\test1\<user name>\logs path.

- NOTE
If any of the user does not have access to the path mentioned in the Data Studio.ini file, then Data Studio closes with the below pop-up message.

---End
The Data Studio.log file is created in the Data Studio\UserData\<user name>\logs path if
- The path is not provided in the Data Studio.ini file.
  For example: -logfolder=.
- The path provided does not exist.

- NOTE
Refer to server manual for detailed information.

You can use any text editor to open and view the Data Studio.log file.

Controlling Exception and Error Logs
The stack trace details of exception, error or throw-able are controlled based on the program argument parameter. This parameter is configured in the Data Studio.ini file.
If the flag value is 'true', then the stack trace details of exception, error or throwable will be saved in the log file.

If the flag value is 'false', then no stack trace details will be saved in the log file.

Description of the Log Message

The description of the log message is as follows:

When the Data Studio.log file reaches the maximum file size of 10000 KB, it will create a new file and save as Data Studio.log.1 automatically and the logs in Data Studio.log are moved to Data Studio.log.1. When Data Studio.log file reaches the maximum file size again, it will create a new file and save as Data Studio.log.2. The logs in Data Studio.log are moved to Data Studio.log.1 and logs in Data Studio.log1 are moved to Data Studio.log.2. Latest logs are always written in Data Studio.log file. This process continues till Data Studio.log.5 reaches the maximum file size and the cycle restarts.

NOTE

To enable performance logging in the server log file, the configuration parameter log_min_messages must be enabled and value must be set as debug1 in the configuration file data/postgresql.conf, that is, log_min_messages = debug1.

Different Types of Log Level

The different types of log levels that are displayed in the Data Studio.log file are as follows:

- **TRACE**: The TRACE level provides detailed information than the DEBUG level.
- **DEBUG**: The DEBUG level indicates the granular information events that are most useful for debug an application.
- **INFO**: The INFO level indicates the information messages that highlight the progress of the application.
- **WARN**: The WARN level indicates potentially harmful situations.
- **ERROR**: The ERROR level indicates error events.
- **FATAL**: The FATAL level indicates event(s) which cause the application to abort.
- **ALL**: The ALL level turns on all the log levels.
- **OFF**: The OFF level turns off all the log levels. This is opposite to ALL level.

NOTE

- If the user enters an invalid value to log level, then log level will be set to WARN.
- If the user does not provide any log level, then log level will be set to WARN.

The logger outputs all messages greater than or equal to its log level. The order of the standard log4j levels are as follows:
### Table 4-4 Logging Level

<table>
<thead>
<tr>
<th></th>
<th>FATAL</th>
<th>ERROR</th>
<th>WARN</th>
<th>INFO</th>
<th>DEBUG</th>
<th>TRACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>FATAL</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>ERROR</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>WARN</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>INFO</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>DEBUG</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>TRACE</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ALL</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

✓ - Log file created ✗ - Log file not created

### 4.3 Getting Started

This section describes the steps to be followed to start Data Studio.

**Prerequisites**

The **StartDataStudio.bat** batch file checks the version of Operating System (OS), Java and Data Studio as a prerequisite to run the Data Studio.

**Step 1** In the **Release package** navigate to Tools folder, locate and double-click **StartDataStudio.bat** to execute and check Java version compatibility.

The batch file checks the version compatibility and will launch Data Studio or display appropriate message based on OS, Java and Data Studio version installed.

If the Java version installed is below 1.8, then appropriate **error message** is displayed.

The scenarios checked by the batch file to confirm the required versions of the OS and Java for DS.

<table>
<thead>
<tr>
<th>DS Installation (32/64 bit)</th>
<th>OS (bit)</th>
<th>Java (bit)</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>32</td>
<td>32</td>
<td>Launches Data Studio</td>
</tr>
<tr>
<td>32</td>
<td>64</td>
<td>32</td>
<td>Launches Data Studio</td>
</tr>
<tr>
<td>DS Installation (32/64 bit)</td>
<td>OS (bit)</td>
<td>Java (bit)</td>
<td>Outcome</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------</td>
<td>-----------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>32</td>
<td>64</td>
<td>64</td>
<td>Error message is displayed</td>
</tr>
<tr>
<td>64</td>
<td>32</td>
<td>32</td>
<td>Error message is displayed</td>
</tr>
<tr>
<td>64</td>
<td>64</td>
<td>32</td>
<td>Error message is displayed</td>
</tr>
<tr>
<td>64</td>
<td>64</td>
<td>64</td>
<td>Launches Data Studio</td>
</tr>
</tbody>
</table>

--- End

### 4.4 Data Studio User Interface

This section describes the user interface of Data Studio.

The Data Studio user interface contains the following:

1. **Main Menu** provides basic operations.
2. **Toolbar** contains buttons for easy access to frequently used operations.
3. **SQL Terminal** tab is used to execute SQL statements and functions/procedures.
4. **PL/SQL Viewer** tab displays the content of functions/procedures.
5. **Editor Area** is used to perform edit operations.
6. **Callstack** pane shows the execution stack.
7. **Breakpoints** pane shows any breakpoints that have been set.
8. **Variables** pane shows variables and their values.
9. **SQL Assistant** tab displays suggestion or reference for the information entered in the SQL Terminal and PL/SQL Viewer.
10. **Result** tab displays the result(s) of an executed function/procedure, or an SQL statement.
11. **Messages** tab displays the output of a process, such as standard input, standard output, and standard error(s).
12. **Object Browser** contains a hierarchical tree display of database connection(s) and related database objects to which the user has access. All default created schemas except for public are grouped under **Catalogs** and user schemas are grouped under **Schemas** below the respective database.
13. **Minimized Window Panel** is used to open Callstack, Breakpoints and Variables pane. This panel is displayed only when Callstack, Breakpoints or Variables pane or all three are minimized.
14. **Search Toolbar** is used to search objects from the Object browser.
4.5 Data Studio Menus

4.5.1 File

The **File** menu contains database connection options. Click **File** from main menu or press **Alt+F** to open the **File** menu.

<table>
<thead>
<tr>
<th>Function</th>
<th>Button</th>
<th>Shortcut Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Connection</td>
<td><img src="image" alt="New Connection" /></td>
<td>Ctrl+N</td>
<td>Creates and adds a new database connection to the Object Browser and SQL Terminal.</td>
</tr>
<tr>
<td>Remove Connection</td>
<td><img src="image" alt="Remove Connection" /></td>
<td>-</td>
<td>Deletes the selected database connection from the Object Browser.</td>
</tr>
<tr>
<td>Connect To DB</td>
<td><img src="image" alt="Connect To DB" /></td>
<td>-</td>
<td>Connects to the database.</td>
</tr>
<tr>
<td>Disconnect From DB</td>
<td><img src="image" alt="Disconnect From DB" /></td>
<td>Ctrl+Shift+D</td>
<td>Disconnects from the selected database.</td>
</tr>
<tr>
<td>Disconnect All</td>
<td><img src="image" alt="Disconnect All" /></td>
<td>-</td>
<td>Disconnects all the databases of a specified connection.</td>
</tr>
<tr>
<td>Open</td>
<td><img src="image" alt="Open" /></td>
<td>Ctrl+O</td>
<td>Loads SQL queries into the SQL Terminal.</td>
</tr>
<tr>
<td>Save</td>
<td><img src="image" alt="Save" /></td>
<td>Ctrl+S</td>
<td>Saves the SQL scripts of the SQL Terminal in an SQL file.</td>
</tr>
<tr>
<td>Save As</td>
<td><img src="image" alt="Save As" /></td>
<td>CTRL+ALT+S</td>
<td>Saves the SQL scripts of the SQL Terminal in an new SQL file.</td>
</tr>
</tbody>
</table>
Closing Data Studio

Follow the steps to close Data Studio:

Step 1 Click the button.
Alternatively choose File > Exit.

Exit Application dialog box is displayed prompting you to take the required action.

Step 2 Click the appropriate buttons based on your requirement.

- **Force Exit** button - To exit the application without saving the SQL History information.

  ☑️ **NOTE**

  Clicking on Force Exit button might not save the SQL History, if not saved yet.

- **Graceful Exit** button - To exit the application after saving the SQL History information to disk in case the save is not complete at this point of time.

- **Cancel** button - To cancel exiting from the application.

----End

4.5.2 Edit

The Edit menu contains clipboard, Format, Find and Replace, and Search Objects operations to use in the PL/SQL Viewer and SQL Terminal tab. Press Alt +E to open the Edit menu.

<table>
<thead>
<tr>
<th>Function</th>
<th>Button</th>
<th>Shortcut Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut</td>
<td><img src="image" alt="Cut" /></td>
<td>Ctrl+X</td>
<td>Cuts the selected text</td>
</tr>
<tr>
<td>Copy</td>
<td><img src="image" alt="Copy" /></td>
<td>Ctrl+C</td>
<td>Copies the selected text or qualified object name</td>
</tr>
<tr>
<td>Paste</td>
<td><img src="image" alt="Paste" /></td>
<td>Ctrl+V</td>
<td>Pastes the selected text or qualified object name</td>
</tr>
<tr>
<td>Format</td>
<td><img src="image" alt="Format" /></td>
<td>Ctrl+Shift+F</td>
<td>Formats all SQL statements and functions/procedures.</td>
</tr>
<tr>
<td>Function</td>
<td>Button</td>
<td>Shortcut Key</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Select All</td>
<td>-</td>
<td>Ctrl+A</td>
<td>Selects all the text in SQL Terminal</td>
</tr>
<tr>
<td>Find and Replace</td>
<td>📦</td>
<td>Ctrl+F</td>
<td>Finds and replaces text in SQL Terminal</td>
</tr>
<tr>
<td>Search Objects</td>
<td>🔍</td>
<td>Ctrl+Shift+S</td>
<td>Searches for objects within a connected database.</td>
</tr>
<tr>
<td>Undo</td>
<td>🔄</td>
<td>Ctrl+Z</td>
<td>Undoes the previous operation</td>
</tr>
<tr>
<td>Redo</td>
<td>🔄</td>
<td>Ctrl+Y</td>
<td>Redoes the previous operation</td>
</tr>
<tr>
<td>UPPERCASE</td>
<td>⌃</td>
<td>Ctrl+Shift+U</td>
<td>Changes the case of the selected text to uppercase</td>
</tr>
<tr>
<td>lowercase</td>
<td>⌃</td>
<td>Ctrl+Shift+L</td>
<td>Changes the case of the selected text to lowercase</td>
</tr>
<tr>
<td>Go To Line</td>
<td>📅</td>
<td>Ctrl+G</td>
<td>Skips to a specific line in the Terminal or PL/SQL Viewer</td>
</tr>
<tr>
<td>Comment/Uncomment Lines</td>
<td>-</td>
<td>Ctrl+/</td>
<td>Comments/Uncomments each selected line</td>
</tr>
<tr>
<td>Comment/Uncomment Block</td>
<td>-</td>
<td>Ctrl+Shift+/</td>
<td>Comments/Uncomments all selected lines or a selected block</td>
</tr>
</tbody>
</table>

**Copy**

Copy can also be used to copy objects from Object Browser.

The format of copied object name is:

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Copied Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functions/Procedures</td>
<td>schema.object name (parameter name parameter type,…)</td>
</tr>
<tr>
<td>Databases</td>
<td>object name</td>
</tr>
<tr>
<td>Schemas</td>
<td>object name</td>
</tr>
<tr>
<td>Tablespaces</td>
<td>object name</td>
</tr>
<tr>
<td>Columns</td>
<td>object name</td>
</tr>
<tr>
<td>Constraints</td>
<td>object name</td>
</tr>
<tr>
<td>Partition names</td>
<td>object name</td>
</tr>
<tr>
<td>All other object types</td>
<td>schema.object name</td>
</tr>
</tbody>
</table>
Search Objects

Search Objects option allows you to search for object(s) from the Object Browser based on the search criteria entered. The search operation can be executed either from **Edit > Search Objects** menu or by clicking the search icon from the Object Browser toolbar. The result of search displays tree structure similar to Object Browser. Right-click operations except for **Refresh** can be performed on the search result objects. Modified objects as a result of drop, set schema, rename, and so on can be viewed only from the main Object Browser after refresh. Right-click options on group names (tables, schema, views and so on) are not allowed on search result objects. Only objects to which you have access can be searched. Objects that you do not have access do not appear in the **Search Scope**.

**NOTE**

Newly added objects can be viewed in the **Search** window by clicking the refresh option at the end of the object type.

![Search Result](image)

Supported Search Options:

<table>
<thead>
<tr>
<th>Search Options</th>
<th>Search Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contains</td>
<td>A search text which contains the searched characters will be displayed.</td>
</tr>
<tr>
<td>Starts With</td>
<td>A search text which starts with the searched character will be displayed.</td>
</tr>
<tr>
<td>Exact Word</td>
<td>A search text which matches exactly with searched characters will be displayed.</td>
</tr>
</tbody>
</table>
A search text with regular expression searches for similar pattern in Object Browser that fulfills the search condition. Select Regular Expression from Search Criteria drop-down to perform this search. For more information, refer to POSIX Regular Expressions rules.

**Example:**
- `^a`, this displays all objects that start with the letter a.
- `^[^A-Za-z]+$`, this displays all objects that do not have alphabets in them.
- `^[^0-9]+$`, this displays all objects that do not have numbers in them.
- `^[a-t][^r-z]+$`, this displays all objects whose name starts between a and t and excludes those that have characters between r and z in them.
- `^e.*a$`, this displays all objects that starts with the letter e and ends with letter a.
- `^[a-z]+$` and select Match Case, this displays all objects that contains only alphabets in lower case.
- `^[A-Z]+$` and select Match Case, this displays all objects that contains only alphabets in upper case.
- `^[A-Za-z]+$` and select Match Case, this displays all objects that contains only alphabets in lower case and upper case.
- `^[A-Za-z0-9]+$` and select Match Case, this displays all objects that contains only alphabets in lower case, upper case and numbers.
- `^".*"$`, this displays all objects that are within in quotes.

**Underscore and % search:**
A search text with _ (underscore) in it considers the underscore as a wildcard of single character. This does not apply to regular expression, starts with and exact word search.

**Example:**
- _ed, this displays all objects that starts with any single character followed by "ed".
- D_t_e, this displays all objects that has character "d", followed by any single character, followed by character "t", followed by any single character, and followed by character "e".

A search text with % (percentage) in it considers the percentage as a wildcard of multiple characters. This does not apply to regular expression, starts with and exact word search.

**Example:**
- %ed, this displays all objects that has "ed" pattern in it.
- D%t%e, this displays all objects that has character "d", followed by any number of characters, followed by character "t", followed by any number of characters, and followed by character "e".

Match case runs the search to match with the search text case.

### 4.5.3 Run

The Run menu contains options to execute a database object in the **PL/SQL Viewer** tab and to execute SQL statements in the **SQL Terminal** tab. Press Alt+R to open the Run menu.

<table>
<thead>
<tr>
<th>Function</th>
<th>Button</th>
<th>Shortcut Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute DB Object</td>
<td></td>
<td>Ctrl+E</td>
<td>Starts execution (in normal mode) of the specified function/procedure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Displays the result in <strong>Result</strong> tab.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Displays the information on actions performed in <strong>Messages</strong> tab.</td>
</tr>
<tr>
<td>Compile/Execute Statement</td>
<td></td>
<td>Ctrl+Enter</td>
<td>Compiles the function/procedure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Starts execution of SQL statements in the <strong>SQL Terminal</strong> tab.</td>
</tr>
<tr>
<td>Function</td>
<td>Button</td>
<td>Shortcut Key</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Cancel</td>
<td></td>
<td>Shift+Esc</td>
<td>Cancels the executing query. Displays the result in Result tab. Displays the information on actions performed in Messages tab.</td>
</tr>
</tbody>
</table>

### 4.5.4 Debug

The Debug menu contains debug operations to use in the PL/SQL Viewer and SQL Terminal tab. Press Alt+D to open the Debug menu.

<table>
<thead>
<tr>
<th>Function</th>
<th>Button</th>
<th>Shortcut Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debug DB Object</td>
<td>🍀</td>
<td>Ctrl+D</td>
<td>Starts the debugging process.</td>
</tr>
<tr>
<td>Continue</td>
<td>🍀</td>
<td>F9</td>
<td>Continues with debugging.</td>
</tr>
<tr>
<td>Terminate Debugging</td>
<td>🍀</td>
<td>F10</td>
<td>Terminates debugging.</td>
</tr>
<tr>
<td>Step Into</td>
<td>🍀</td>
<td>F7</td>
<td>Steps through the code statement.</td>
</tr>
<tr>
<td>Step Over</td>
<td>🍀</td>
<td>F8</td>
<td>Steps over the function.</td>
</tr>
<tr>
<td>Step Out</td>
<td>🍀</td>
<td>Shift+F7</td>
<td>Steps out of the function.</td>
</tr>
</tbody>
</table>

### 4.5.5 Settings

The Settings menu contains the option to change the language. Press Alt+G to open the Settings menu.

<table>
<thead>
<tr>
<th>Function</th>
<th>Shortcut Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>-</td>
<td>Set the language for Data Studio user interface.</td>
</tr>
<tr>
<td>Preferences</td>
<td>-</td>
<td>Set the user preferences in Data Studio.</td>
</tr>
</tbody>
</table>

### Setting the Language

Follow the steps to modify the language from English to Chinese:
Step 1  Choose Settings > Language > (zh_CN) 中文 (简体) (C).

Restart Data Studio dialog box is displayed.

⚠️ NOTE

Save all data before modifying the language.

Step 2  Click Yes.

If you click No, the language settings are applied the next time you restart Data Studio.

The language is set after you restart Data Studio.

----End

4.5.6 Help

The Help menu contains the user manual and version information of Data Studio. Press Alt + H to open the Help menu.

<table>
<thead>
<tr>
<th>Function</th>
<th>Shortcut Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>About Data Studio</td>
<td>-</td>
<td>Displays the current version and copyright information of Data Studio.</td>
</tr>
</tbody>
</table>

⚠️ NOTE

The Version displayed below is indicative and may not reflect the current version of Data Studio.

![About Data Studio](image)

**Version:** Data Studio 6.5.1  
**Java Version:** 1.8.0_121  
**Java Home:** C:\Program Files (x86)\Java\jre1.8.0_121  
**Build Time:** 2019-06-13 05:23

Copyright © 2016-2018 Huawei Technologies Co., Ltd.  
All Rights Reserved.  
http://www.huawei.com
4.6 Data Studio Toolbars

The following image shows the toolbar:

![Toolbar Image]

The toolbar contains the following actions:

- New Connection
- Remove Connection
- Connect To DB
- Disconnect From DB
- Disconnect All
- Open SQL Script
- Save SQL Script
- Connection Profile Drop-down List
- Open New SQL Terminal
- Execute DB Object
- Debug DB Object
- Compile a Function/Procedure
- Step Into
- Step Out
- Step Over
- Terminate Debugging
- Continue Debugging
- Execution Plan and Cost
- Execution Plan and Cost - Graphical
- Format
- Upper Case
- Lower Case
- SQL Assistant

4.7 Data Studio Right-Click Menus

This section describes the right-click menus of Data Studio.

Object Browser Pane

The following image shows the Object Browser pane:
Right-clicking the connection profile allows you to select **Rename, Edit, Remove Connection, and Properties** along with **Refresh** options.

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>Shortcut Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rename Connection</td>
<td>-</td>
<td>Renames a connection name.</td>
</tr>
<tr>
<td>Edit Connection</td>
<td>-</td>
<td>Modifies connection profile details.</td>
</tr>
<tr>
<td>Remove Connection</td>
<td>-</td>
<td>Removes the existing database connection.</td>
</tr>
<tr>
<td>Properties</td>
<td>-</td>
<td>Shows the properties of the connection.</td>
</tr>
<tr>
<td>Refresh</td>
<td>F5</td>
<td>Refreshes the connection.</td>
</tr>
</tbody>
</table>

Right-clicking the **Databases** group allows you to select **Create Database, Disconnect All, and Refresh** options.

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>Shortcut Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Database</td>
<td>-</td>
<td>Creates a new database in this connection.</td>
</tr>
<tr>
<td>Menu Item</td>
<td>Shortcut Key</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Disconnect All</td>
<td>-</td>
<td>Disconnects all the databases of this connection.</td>
</tr>
<tr>
<td>Refresh</td>
<td>F5</td>
<td>Refreshes the databases group.</td>
</tr>
</tbody>
</table>

Right-clicking the active database allows you to select **Disconnect from DB**, **Open Terminal**, **Properties**, and **Refresh** options.

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>Shortcut Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disconnect from DB</td>
<td>Ctrl+Shift+D</td>
<td>Disconnects the database</td>
</tr>
<tr>
<td>Open Terminal</td>
<td>Ctrl+T</td>
<td>Open a Terminal in this connection</td>
</tr>
<tr>
<td>Properties</td>
<td>-</td>
<td>Displays the properties of the database</td>
</tr>
<tr>
<td>Refresh</td>
<td>F5</td>
<td>Refreshes the database</td>
</tr>
</tbody>
</table>

Right-clicking the inactive database allows you to select **Connect to DB**, **Rename Database**, and **Drop Database** options.

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>Shortcut Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect to DB</td>
<td>-</td>
<td>Connects the database</td>
</tr>
<tr>
<td>Rename Database</td>
<td>-</td>
<td>Renames the database name</td>
</tr>
<tr>
<td>Drop Database</td>
<td>-</td>
<td>Drops the database</td>
</tr>
</tbody>
</table>

Right-clicking the **Catalogs** group allows you to select **Refresh** option.

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>Shortcut Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refresh</td>
<td>F5</td>
<td>Refreshes the schema</td>
</tr>
</tbody>
</table>

Right-clicking the **Schemas** group allows you to select **Create Schema**, **Grant/Revoke** and **Refresh** option.

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>Shortcut Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Schema</td>
<td>-</td>
<td>Creates a new schema</td>
</tr>
<tr>
<td>Grant/Revoke</td>
<td>-</td>
<td>Grant/Revoke access to schema group</td>
</tr>
<tr>
<td>Menu Item</td>
<td>Shortcut Key</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>Refresh</td>
<td>F5</td>
<td>Refreshes the schema</td>
</tr>
</tbody>
</table>

Right-clicking the schema allows you to select **Export DDL, Export DDL and Data, Rename Schema, Drop Schema, Grant/Revoke**, and **Refresh** options.

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>Shortcut Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Export DDL</td>
<td>-</td>
<td>Exports DDL of the schema</td>
</tr>
<tr>
<td>Export DDL and Data</td>
<td>-</td>
<td>Exports DDL and data of the schema</td>
</tr>
<tr>
<td>Rename Schema</td>
<td>-</td>
<td>Renames a schema</td>
</tr>
<tr>
<td>Drop Schema</td>
<td>-</td>
<td>Drops a schema</td>
</tr>
<tr>
<td>Grant/Revoke</td>
<td>-</td>
<td>Grant/Revoke Access to schema</td>
</tr>
<tr>
<td>Refresh</td>
<td>F5</td>
<td>Refreshes the schema</td>
</tr>
</tbody>
</table>

Right-clicking **Functions/Procedures** allows you to select **Refresh** and **Create Function/Procedure** and **Grant/Revoke** options.

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>Shortcut Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create PL/SQL Function</td>
<td>-</td>
<td>Creates PL/SQL Function</td>
</tr>
<tr>
<td>Create PL/SQL Procedure</td>
<td>-</td>
<td>Creates PL/SQL Procedure</td>
</tr>
<tr>
<td>Create SQL Function</td>
<td>-</td>
<td>Creates SQL Function</td>
</tr>
<tr>
<td>Create C Function</td>
<td>-</td>
<td>Creates C Function</td>
</tr>
<tr>
<td>Grant/Revoke</td>
<td>-</td>
<td>Grant/Revoke access to Function/Procedure</td>
</tr>
<tr>
<td>Refresh</td>
<td>F5</td>
<td>Refreshes the Function/Procedure</td>
</tr>
</tbody>
</table>

Right-clicking **Tables** allows you to select **Refresh** and **Create table** and **Grant/Revoke** options.

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>Shortcut Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Regular Table</td>
<td>-</td>
<td>Creates Regular table</td>
</tr>
<tr>
<td>Create Partition Table</td>
<td>-</td>
<td>Creates partition table</td>
</tr>
<tr>
<td>Menu Item</td>
<td>Shortcut Key</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>Grant/Revoke</td>
<td>-</td>
<td>Grant/revokes access of the table</td>
</tr>
<tr>
<td>Refresh</td>
<td>F5</td>
<td>Refreshes the table</td>
</tr>
</tbody>
</table>

Right-clicking **Views** allows you to select **Refresh** and **Create View** and **Grant/Revoke** options.

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>Shortcut Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create View</td>
<td>-</td>
<td>Creates View</td>
</tr>
<tr>
<td>Grant/Revoke</td>
<td>-</td>
<td>Grant/revokes the access of views</td>
</tr>
<tr>
<td>Refresh</td>
<td>F5</td>
<td>Refreshes the View</td>
</tr>
</tbody>
</table>

Right-clicking the **PL/SQL Viewer** allows you to **Cut, Copy, Paste, Select All, Comment/Uncomment Lines, Comment/Uncomment Block, Compile, Execute, Add Variable To Monitor, Debug with Rollback** and **Debug** in the **PL/SQL Viewer** tab.

<table>
<thead>
<tr>
<th>Right-Click Options</th>
<th>Shortcut Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut, Copy, Paste</td>
<td>Ctrl+X, Ctrl+C, Ctrl+V</td>
<td>Clipboard operations</td>
</tr>
<tr>
<td>Select All</td>
<td>Ctrl+A</td>
<td>Selects the content in PL/SQL Viewer</td>
</tr>
<tr>
<td>Comment/Uncomment Lines</td>
<td>-</td>
<td>Comment/Uncomment selected line</td>
</tr>
<tr>
<td>Comment/Uncomment Block</td>
<td>-</td>
<td>Comment/Uncomment selected Block.</td>
</tr>
<tr>
<td>Compile</td>
<td>-</td>
<td>Compiles the function/procedure</td>
</tr>
<tr>
<td>Execute</td>
<td>-</td>
<td>Executes the function/procedure</td>
</tr>
<tr>
<td>Add Variable To Monitor</td>
<td>-</td>
<td>Adds selected variable to the monitor window</td>
</tr>
<tr>
<td>Debug with Rollback</td>
<td>-</td>
<td>Debug Function/Procedure and rollback changes after debug completion.</td>
</tr>
<tr>
<td>Debug</td>
<td>-</td>
<td>Debugs the function/procedure</td>
</tr>
</tbody>
</table>

Right-clicking the **SQL Terminal** allows you to perform following operations:
Right-clicking in the **Messages** tab allows you to **Copy**, **Select All**, and **Clear** the contents of the **Messages** tab.

<table>
<thead>
<tr>
<th>Right-Click Options</th>
<th>Shortcut Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy</td>
<td>Ctrl+C</td>
<td>Copies the text</td>
</tr>
</tbody>
</table>

Right-Click Options

<table>
<thead>
<tr>
<th>Cut</th>
<th>Ctrl+B</th>
<th>Clipboard operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy</td>
<td>Ctrl+C</td>
<td>Copies the text</td>
</tr>
<tr>
<td>Paste</td>
<td>Ctrl+V</td>
<td>Clipboard operations</td>
</tr>
<tr>
<td>Select All</td>
<td></td>
<td>Selects all text</td>
</tr>
<tr>
<td>Execute Statement</td>
<td></td>
<td>Executes query</td>
</tr>
<tr>
<td>Open</td>
<td></td>
<td>Opens file</td>
</tr>
<tr>
<td>Save</td>
<td></td>
<td>Saves the query</td>
</tr>
<tr>
<td>Find and Replace</td>
<td></td>
<td>Finds and replaces text in SQL Terminal</td>
</tr>
<tr>
<td>Execution Plan</td>
<td></td>
<td>Executes the query</td>
</tr>
<tr>
<td>Comment/Uncomment Lines</td>
<td>Ctrl+/</td>
<td>Comment/Uncomment selected lines</td>
</tr>
<tr>
<td>Comment/Uncomment Block</td>
<td>Ctrl+Shift+/</td>
<td>Comment/Uncomment selected block of lines</td>
</tr>
<tr>
<td>Cancel</td>
<td></td>
<td>Cancels execution</td>
</tr>
<tr>
<td>Save As</td>
<td>CTRL+ALT+S</td>
<td>Saves the query in a new file .</td>
</tr>
<tr>
<td>Format</td>
<td>CTRL+SHIFT +F</td>
<td>Formats the selected SQL statements.</td>
</tr>
</tbody>
</table>
### 4.8 Connection Profiles

#### 4.8.1 Overview

When Data Studio is started, the **New Database Connection** dialog box will open by default. To perform any DB operations, Data Studio must be connected to at least one database.

Enter the connection parameters to create a new database connection between Data Studio and the database in the server. Hovering over the connection name will display the server information.

**NOTE**

You need to fill all the mandatory parameters, that are marked with asterisk (*) to complete the operation successfully.

#### 4.8.2 Adding a Connection

Follow the steps to establish a new database connection:

**Step 1** Choose **File > New Connection** from the main menu, or click on the toolbar, or press **Ctrl+N** to connect to the database.

The **New Database Connection** dialog box is displayed.
While establishing a connection, if the preference file is corrupted or the preferences values are invalid, then an error message is displayed informing you that preference values are invalid and default values are set for preferences. To complete establishing a new database connection operation, click OK.

**Step 2** The table on the left lists the details of the existing connection profile(s) used to connect to the database along with the server information.

- Double clicking a connection name populates the connection parameters such as **Connection Name**, **Host**, and **Host Port**.

  - **NOTE**
    - The server information is displayed only after one successful connection.
    - Double clicking a connection name populates the connection parameters such as **Connection Name**, **Host**, and **Host Port**.

  - **NOTE**
    - If password is corrupted for any of the existing connection profile or the key is corrupted, then the password field needs to be filled in for all created connections.

- Clicking ![icon](image) displays different pop-up messages based on the connection status of database.
  - If the database connection is active, then **Remove Connection Confirmation** pop-up is displayed. Click **Yes** to disconnect all databases.
  - If the database connection is not active, then **Remove Connection Confirmation** pop-up is displayed.

- Clicking ![icon](image) without a connection name displays a pop up stating to select at least one connection profile.

**Step 3** Provide the following credentials to enter a new set of parameters to connect to the database:

- **NOTE**
  - You can click **Clear** to clear all fields in the **New Database Connection** dialog box.
  - Use shortcut key (Ctrl+V) to paste data in Connection window. Data Studio does not support right-click options for all dialog boxes.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Type</td>
<td>Select the database type.</td>
<td>Analytical clusterDWS</td>
</tr>
<tr>
<td>Connection Name</td>
<td>Provide a connection name.</td>
<td>My_Connection_DB</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Host</td>
<td>Provide the IP address (IPv4) or server domain name.</td>
<td>db.dws.mycloud.com</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong></td>
<td>10.xx.xx.xx</td>
</tr>
<tr>
<td></td>
<td>• If domain name length is greater than 25 characters, then the complete domain name will not be displayed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>test1(db.dws...com: 25xxx)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Hovering over the connection name once the connection is established will show the server IP and version.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Entry made in this field will be validated for IP address if it has format of digits with three separators (.). Any entry not meeting this validation will be considered as domain name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• A typical domain name:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Starts with a letter.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Allows letters, digits, hyphens (-), and period (.). All other special characters are not allowed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Does not allow space/tabs.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Length cannot exceed 253 characters and a maximum of 63 characters is allowed in between periods.</td>
<td></td>
</tr>
<tr>
<td>Host Port</td>
<td>Provide the port address.</td>
<td>25001</td>
</tr>
<tr>
<td>Database Name</td>
<td>Provide the database name.</td>
<td>postgres</td>
</tr>
<tr>
<td>User Name</td>
<td>Provide the user name to connect to the selected database.</td>
<td>-</td>
</tr>
</tbody>
</table>
**Table:**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password</td>
<td>Provide the password to connect to the database. The password text is masked.</td>
<td>-</td>
</tr>
</tbody>
</table>

- Select an option from the **Save Password** drop-down list. The options available are:
  - **Permanently**: Saves the password even after exiting Data Studio. While establishing the connection for the first time this option will not be available. Refer to the **Save Password Permanently** section for information to hide/view this drop-down option.
  - **Current Session Only**: Saves the password only for the current session.
  - **Do Not Save**: Does not save the password. If this option is selected, Data Studio prompts for the password for certain operations such as:
    - Creating a Database
    - Renaming a Database
    - Debugging a PL/SQL Function
    - Working with the SQL Terminals
- **Enable SSL** check box is selected by default.

**Step 4** Follow the steps to enable SSL:
1. Select the **Enable SSL** option.
2. Click the **SSL** tab.
3. Provide the following information. The following files are required for secured connection. Refer to **SSL Certificates** section.
   - To select the **Client SSL Certificate**, click ![ClientSSL](image) and select the Client SSL Certificate.
- To select the **Client SSL Key**, click ![ ] and select the Client SSL key.
- To select the **Root Certificate**, click ![ ] and select the Root Certificate.
- Select the Secure Sockets Layer (SSL) Mode from **SSL Mode** drop-down. Refer to following table for description of different SSL modes:

<table>
<thead>
<tr>
<th>SSL Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>require</td>
<td>Selecting require will not check the validity of the certificates since a non-validating SSL factory is used.</td>
</tr>
<tr>
<td>verify-ca</td>
<td>Selecting verify-ca checks if the ca is correct using a validating SSL factory.</td>
</tr>
<tr>
<td>verify-full</td>
<td>Selecting verify-full checks if the ca and host is correct using a validating SSL factory.</td>
</tr>
</tbody>
</table>

---

**NOTE**

- If the **SSL Mode** is selected as verify-ca or verify-full, then it is mandatory to select the Root Certificate.
- DS prompt for the Client key while accessing the **gs_dump** feature for the first time.

**NOTE**

- Selecting **Client SSL Certificate** and **Client SSL Key** ensures secured connection for export of DDL and data using Data Studio.
- Selecting invalid file for Client SSL Certificate and/or Client SSL Key will result in export failure. Refer to **Troubleshooting**.
- If you deselect **Enable SSL** check box and proceed, then **Connection Security Alert** dialog box is displayed. Refer to **Security Disclaimer** for information to display this security alert or not.
  - **Continue** - Clicking Continue proceeds with insecure connection.
  - **Cancel** - Clicking Cancel proceeds to enable SSL.
  - **Do not show again** - Checking this field hides the **Connection Security Alert** dialog box for subsequent connections for current logged instance of Data Studio.
- Refer to server manual for detailed information.

**Step 5** Follow the steps to set the **Fast Load Options**:

1. Click the **Advanced** tab.
2. Enter the schema names using comma separator to load on priority while establishing a connection in the **Include** field.

3. Enter the schema names using comma separator to avoid loading on priority while establishing a connection in the **Exclude** field.

4. Select an option from the **Load Objects** options. The options available are:
   - **All Objects** - Loads all objects.
   - **Objects allowed as per user privilege** - Loads only objects that the user has access. Refer to **Minimum Privileges Requirement** table for the minimum access required for objects to be listed in Object Browser.

   **NOTE**
   
The default value is **Objects allowed as per user privilege**.

5. Enter the load limit in **Load Limit** field. The maximum value allowed is 30000. This is the database object count.

   **NOTE**
   
   - If the number of object type (tables, view..) of the schema mentioned in the **Include** field is greater than the value entered in the **Load Limit** field, then only the parent objects for a schema will be loaded. This implies that child objects like columns, constraints, indexes, functions with more than three parameters, and so on will not be loaded.
   - Schema names provided in the Include and Exclude lists are validated.
   - If you do not have access to the schema name entered in the **Include** field, then an appropriate error message is displayed for that schema during connection.
   - If you do not have access to the schema name entered in the **Exclude** field, then the schema will not be loaded in **Object Browser** after connection is established.

**Step 6** Click **OK** to establish the connection successfully.

The status bar displays the status of the completed operation.

While Data Studio is connecting to the database, the following status bar shows the status:

![Connecting to server](image)

Once the connection is established, all schema objects will be displayed in the **Object Browser** pane.
Data Studio allows you to login even if the password has expired with a message informing that some operations may not work as expected. Refer to Password Expiry for information to modify this behavior.

- Refer to Cancel Connection section to cancel the connection.
- Postgres specific schemas are not displayed in the Object Browser.

Cancel Connection

Follow the steps to cancel the connection operation:

**Step 1** Click Cancel.

The Cancel Connection dialog box is displayed.

**Step 2** Click Yes.

A message confirmation dialog box is displayed.

**Step 3** Click OK.

Lazy Loading

Lazy loading feature delays the loading of objects until required.

When you connect to a database only child objects of schema saved under search_path will be loaded as shown below:

Unloaded schemas are represented as "schema (…)".
To load child objects expand the schema. During expansion of schema, the objects under the schema will show as loading:

**NOTE**

If you try to load an unloaded object while loading is in progress for another object, a pop-up message is displayed informing you that another loading is in progress. The icon next to the unloaded object disappears. Refresh at the object or database level to display this icon again for loading.

Expand schema to load and view the child objects. The Object Browser can load child objects of only one schema at a time.

If `search_path` is modified after establishing connection, then the changes will be reflected only after reconnecting the database. Auto-suggest works on schema names, table names, views, and table name aliases for all schema objects that you have access.

A maximum of 50,000 objects are loaded in the **Object Browser** pane within one minute.

Database connection timeout is set as three minutes (or 180 seconds) by default, beyond which connection time out error is displayed.

You can set the `loginTimeout` value in `Data Studio.ini` file. The file is present in the `Data Studio|Data Studio.ini` path.
When user log in to the Data Studio, `pg_catalog` is loaded automatically.

### 4.8.3 Renaming a Connection

Follow the steps to rename a database connection:

**Step 1** In the **Object Browser** pane, right-click the selected connection name and select **Rename Connection**.

A **Rename Connection** dialog box is displayed prompting you to provide the new name for the connection.

**Step 2** Enter the new connection name. Select the **OK** to rename the connection.

The status bar displays the status of the completed operation.

*NOTE*  
The new connection name must be unique else the rename operation fails.

### 4.8.4 Editing a Connection

Follow the steps to edit the database connection properties:

**Step 1** In the **Object Browser** pane, right-click the selected connection name and select **Edit Connection**.

Editing an active connection will require closing the connection and then reopening the connection with the new properties. A warning message about connections being reset is shown.

The **Edit Connection** dialog box is displayed.

**Step 2** Click **OK** to proceed or **Cancel** to exit the operation.

*NOTE*  
The **Connection Name** field cannot be modified.

**Step 3** Edit the connection parameters. Connection parameters are explained in **Adding a Connection**.

**Step 4** Click **OK** to save the updated connection information.

*NOTE*

- You can click **Clear** to clear all fields in the **Edit Database Connection** dialog box.
- If you click **OK** without modifying any of the connection parameters, no changes saved dialog message is displayed. Until a connection parameter is changed the dialog message is displayed.
- Data Studio allows you to login even if the password has expired with a message informing that some operations may not work as expected. Refer to **Password Expiry** for information to change this behavior.
- Refer to **Cancel Connection** section to cancel the connection.

If SSL is not enabled, then a **Connection Security Alert** dialog box is displayed.
Step 5 Click **Continue** to proceed with insecured connection or click **Cancel** to return to the **Edit Connection** dialog to enable SSL.

---

**NOTE**

_Do not show again_ option is used to hide the **Connection Security Alert** dialog box for subsequent connections.

The **Remove Server Confirmation** is displayed to confirm closing databases for the edited connection.

Step 6 Click **Yes** to proceed to updating the connection information and reconnecting the connection with the updated parameters.

The status bar displays the status of the completed operation.

---End

### 4.8.5 Removing a Connection

Follow the steps to remove an existing database connection:

Step 1 Right-click the selected connection name and select **Remove Connection**.

A confirmation dialog box is displayed to remove the connection.

Step 2 Click **Yes** to remove the server connection.

The status bar displays the status of the completed operation.

This action will remove the connection from the **Object Browser**. Any unsaved data will be lost.

---End

### 4.8.6 Viewing Connection Properties

Follow the steps to view the properties of a connection:

Step 1 Right-click the selected connection and select **Properties**.

The status bar displays the status of the completed operation.

Properties of the selected connection is displayed.

---

**NOTE**

If the property of a connection is modified for the connection that is already opened, then open the properties of the connection again to view the updated information on the same opened window.

---End

### 4.8.7 Refreshing a Database Connection

Follow the steps to refresh the database connection:

Step 1 In the **Object Browser** pane, right-click the selected connection name and select **Refresh** or press **F5**.
The status bar displays the status of the completed operation.

----End

The time taken to refresh the database depends on the number of objects present in the database. For a large database, it is recommended that perform this operation only if required.

- If you right-click the connection name and select **Refresh**, the connection profile is refreshed. During refresh, the connection will be updated with the latest copy from the server.
- If you right-click the Schema and select **Refresh**, all functions/procedures and tables under the schema are refreshed. During refresh, all functions/procedures and tables are updated with the latest copy from the server.
- If any stored function/procedure is deleted from the database before the refresh operation, then it will be removed from the **Object Browser** only after you perform the refresh operation.
- If you right-click a specific function/procedure and select **Refresh**, the specific function/procedure is refreshed. During refresh, the specific function/procedure is updated with the latest copy from the server.
- If you refresh at database level or connection profile level, then all the child objects of schema in **search_path** along with the schema already expanded by the user will be loaded.
- If you re-connect to the Database, then only schema objects saved under **search_path** will be loaded. Previously expanded objects will not be loaded.
- Database and multiple objects under a database cannot be refreshed simultaneously.

**Exporting/Importing Connection Details**

Data Studio provides the option to export/import connection details from the connection dialog for future reference.

Following fields are exported:

- SSL Mode
- Connection name
- Server IP
- Server Port
- Database Name
- Username
- clSSLCertificatePath
- clSSLLKeyPath
- profileId
- rootCertPath
- connectionDriverName
- schemaExclusionList
- schemaInclusionList
Follow the steps to access the feature:

**Step 1** Click **File** on Menu Bar. Following window is displayed:

![Data Studio File Menu](image)

**Step 2** Select **Export Connections** to export the connection profiles. **Export Connection Profiles** window is displayed to the user to select the connections which needs to be exported.
Select the connections you want to export and enter a file name where the exported connections will be saved. Click OK.

Select the location where you want to save the file and Click OK.

A dialog box is displayed once the connections are exported successfully.
Step 3 To import the connection profiles select **Import Connections**.

Step 4 Select the file you want to import and click **Open**.

If there is any match between the connections being imported and the existing connections, a dialog box is displayed.

- **Replace** - The imported connection profile will be replaced with the existing one.
- **Copy, but keep both files** - The imported connection profile will be renamed.
- **Don't Copy** - The existing connection profile will remain as it was.
- **Do this for all conflicts** - Same action will be repeated for all the matches.
Click any of the provided options according to the scenario and click OK

---End

**NOTE**

Password and SSL password field are not be exported.

### 4.9 Databases

#### 4.9.1 Creating a Database

A relational database is a database that has a set of tables which is manipulated in accordance with the relational model of data. It contains a set of data objects used to store, manage, and access data.

Examples of such data objects are tables, views, indexes, functions and so on.

Follow the steps to create a database:

**Step 1** In the **Object Browser** pane, right-click the selected **Databases** group and select **Create Database**.

**NOTE**

This operation can be performed only when there is at least one active database.

A **Create Database** dialog box is displayed prompting you to provide the necessary information to create the database.

**Step 2** Enter the database name. Refer to server manual for database naming rules.

**Step 3** Select the required type of encoding character set from the **Database Encoding** drop-down list.

The database supports **UTF-8**, **GBK**, **SQL_ASCII**, and **LATIN1** types of encoding character sets. Creating the database with other encoding character sets may result in erroneous operations.

**Step 4** Select the **Connect to the DB** check box and click **OK**.

The status bar displays the status of the completed operation.

You can view the created database in the **Object Browser**. The system related schema present in the server is automatically added to the new database.

**NOTE**

Data Studio allows you to login even if the password has expired with a message informing that some operations may not work as expected when no other database is connected in that connection profile.

Refer to **Password Expiry** for information to change this behavior.

---End

**Cancel Connection**

Follow the steps to cancel the connection operation:
Step 1 Double-click the status bar to open the Progress View tab.

Step 2 In the Progress View tab, click ⌠.

Step 3 In the Cancel Operation dialog box, click Yes.

The status bar displays the status of the cancelled operation.

----End

4.9.2 Disconnecting All Databases

You can disconnect all the databases from a connection.

Follow the steps below to disconnect a connection from the database:

Step 1 In the Object Browser pane, right-click the selected Databases group and select Disconnect All. This will disconnect all the databases under that connection.

NOTE

This operation can be performed only when there is at least one active database.

A confirmation dialog box is displayed to disconnect all databases for the connection.

Step 2 Click Yes to disconnect.

The status bar displays the status of the completed operation.

Data Studio populates all the connection parameters (except password) that were provided during the last successful connection with the database. To reconnect, you need to enter only the password in the connection wizard.

----End

4.9.3 Connecting to Database

You can connect to the database.

Follow the steps to connect a database:

Step 1 In the Object Browser pane, right-click the selected database name and select Connect to DB.

NOTE

This operation can be performed only on an inactive database.

The database is connected.

The status bar displays the status of the completed operation.
Data Studio allows you to login even if the password has expired with a message informing that some operations may not work as expected when no other database is connected in that connection profile. Refer to Password Expiry for information to change this behavior.

- Refer to Cancel Connection section to cancel the connection to database.

4.9.4 Disconnecting Database

You can disconnect the database.

Follow the steps to disconnect a database:

Step 1 In the Object Browser pane, right-click the selected database name and select Disconnect from DB.

NOTE

This operation can be performed only on an active database.

A confirmation dialog box is displayed to disconnect database.

Step 2 Click Yes to disconnect.

The database is disconnected.

The status bar displays the status of the completed operation.

4.9.5 Renaming a Database

Follow the steps to rename a database:

Step 1 In the Object Browser pane, right-click the selected database and select Rename Database.

NOTE

This operation can be performed only on an inactive database.

A Rename Database dialog box is displayed prompting you to provide the necessary information to rename the database.

Step 2 Enter the new database name. Select the Connect to the DB? check box and click OK.

A confirmation dialog box is displayed to rename the database.

Step 3 Click OK to rename the database.

The status bar displays the status of the completed operation.

You can view the renamed database in the Object Browser.
Refer to Cancel Connection section to cancel the connection to database.

4.9.6 Dropping a Database

Individual or batch drop can be performed on databases. Refer to Dropping Batch of Objects section for batch drop.

Follow the steps to drop a database:

Step 1 In the Object Browser pane, right-click the selected database and select Drop Database.

NOTE

This operation can be performed only on an inactive database.

A confirmation dialog box is displayed to drop the database.

Step 2 Click OK to drop the database.

A popup message and status bar displays the status of the completed operation.

4.9.7 Viewing a Database Properties

Follow the steps to view the properties of a database:

Step 1 Right-click the selected database and select Properties.

NOTE

This operation can be performed only on an active database.

The status bar displays the status of the completed operation.

The properties of the selected database is displayed.

NOTE

If the property of a database is modified for the database that is already opened, then refresh and open the properties of the database again to view the updated information on the same opened window.

4.10 Schemas

4.10.1 Overview

This section describes working with database schemas. All system schemas are grouped under Catalogs and user schemas under Schemas.
4.10.2 Creating a Schema

In relational database technology, schemas provide a logical classification of objects in the database. Some of the objects that a schema may contain include functions/procedures, tables, sequences, views, and indexes.

Follow the steps below to define a schema:

**Step 1** In the Object Browser pane, right-click the selected Schemas group and select Create Schema.

⚠️ NOTE

Only refresh can be performed on Catalogs group.

**Step 2** Enter the schema name and click OK. You can create the schema only if the database connection is active.

You can view the new schema in the Object Browser pane.

The status bar displays the status of the completed operation.

----End

You can perform the following actions on a schema:

- Refresh a Schema - To refresh a schema, right-click the selected Schema Name and select Refresh Schema. All the objects under that schema will be refreshed.
- Rename Schema (Refer to Renaming a Schema for more details)
- Drop Schema (Refer to Dropping a Schema for more details)
- Export Data Definition Language (DDL) (Refer to Exporting Schema DDL for more details)
- Export DDL and Data (Refer to Exporting Schema DDL and Data for more details)
- Grant/Revoke Privilege (Refer to Grant/Revoke Privilege for more details)

**Displaying Default Schema**

Data studio displays default schema of the user in the toolbar.

When a create query without mentioning the schema name is executed from SQL Terminal, the corresponding objects are created under the default schema of the user.
When a select query is executed in SQL terminal without mentioning the schema name, default schemas are searched to find these objects.

When Data Studio starts, the default schemas are set to <username>, public schemas in same priority.

If another schema is selected in drop down, selected schema will be set as default schema, overriding previous setting.

The selected schema is set as default schema for all active connections of the database (selected in database list drop down).

**NOTE**

This feature is not available for OLTP database.

### 4.10.3 Exporting Schema DDL

Exporting the schema DDL exports the DDLs of functions/procedures, tables, sequences and views of the schema.

Follow the steps to export the schema DDL:

**Step 1** In the **Object Browser** pane, right-click the selected schema and select **Export DDL**.

The **Data Studio Security Disclaimer** dialog box is displayed. You can turn off this security disclaimer message. Refer to **Security Disclaimer** section for more information.

**Step 2** Click **OK**.

The **Save As** dialog box is displayed.

**Step 3** In the **Save As** dialog box, select the location to save the DDL and click **Save**. The status bar displays the progress of the operation.
To cancel the export operation, double-click the status to open the Progress View tab and click \( \checkmark \) to close the operation. For more information refer to Canceling the export table data operation.

The exported file name will not be the same as schema name, if the schema name contains characters which are not supported by Windows.

Microsoft Visual C runtime file (msvcr100.dll) is required to complete this operation. Refer to Troubleshooting section for more information.

The Export message and status bar displays the status of the completed operation.

<table>
<thead>
<tr>
<th>Database Encoding</th>
<th>File Encoding</th>
<th>Supports Exporting DDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTF-8</td>
<td>UTF-8</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>GBK</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>LATIN1</td>
<td>Yes</td>
</tr>
<tr>
<td>GBK</td>
<td>GBK</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>UTF-8</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>LATIN1</td>
<td>No</td>
</tr>
<tr>
<td>LATIN1</td>
<td>LATIN1</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>GBK</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>UTF-8</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Multiple objects can be selected to export DDL. Refer to Batch Export section for list of objects not supported for export DDL operation.

4.10.4 Exporting Schema DDL and Data

Exporting the schema DDL and data exports the following:

- DDLs of functions/procedures of the schema.
- DDLs and data of tables of the schema.
- DDLs of views of the schema.
- DDLs of sequences of the schema.

Follow steps to export the schema DDL and data:

Step 1 In the Object Browser pane, right-click the selected schema and select Export DDL and Data.
The **Data Studio Security Disclaimer** dialog box is displayed. You can turn off this security disclaimer message. Refer to **Security Disclaimer** section for more information.

**Step 2** Click **OK**.

The **Save As** dialog box is displayed.

**Step 3** In the **Save As** dialog box, select the location to save the DDL and data and click **Save**. The status bar displays the progress of the operation.

**NOTE**
- To cancel the export operation, double-click the status to open the **Progress View** tab and click ❌. For more information, refer to **Canceling the export table data operation**.
- The exported file name will not be the same as schema name, if the schema name contains characters which are not supported by Windows.
- Microsoft Visual C runtime file (msvcrt100.dll) is required to complete this operation. Refer to **Troubleshooting** section for more information.

The **Export** message and status bar displays the status of the completed operation.

<table>
<thead>
<tr>
<th>Database Encoding</th>
<th>File Encoding</th>
<th>Supports Exporting DDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTF-8</td>
<td>UTF-8</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>GBK</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>LATIN1</td>
<td>Yes</td>
</tr>
<tr>
<td>GBK</td>
<td>GBK</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>UTF-8</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>LATIN1</td>
<td>No</td>
</tr>
<tr>
<td>LATIN1</td>
<td>LATIN1</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>GBK</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>UTF-8</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**NOTE**

Multiple objects can be selected to export DDL and Data. Refer to **Batch Export** section for list of objects not supported for export DDL and Data operation.

----End

4.10.5 Renaming a Schema

Follow the steps below to rename a schema:

**Step 1** In the **Object Browser** pane, right-click the selected schema and select **Rename Schema**.
Step 2 Enter the schema name and click OK. You can view the renamed schema in the Object Browser. The status bar displays the status of the completed operation.

----End

4.10.6 Support Sequence DDL

Data Studio provides the option to show sequence DDL or allow users to export sequence DDL. It provides "Show DDL", "Export DDL", "Export DDL and Data"

Follow the steps to access the feature:

Step 1 In Object Browser right-click on any object under Sequences. A menu option will appear

Step 2 Select Show DDL option to see the DDL statements. Or Select Export DDL menu option to export DDL statements. Or Select Export DDL and Data menu option to export DDL statements and the select statement.

Refer to the following image:

![Sequence Browser with options]

☐ NOTE

If user is sequence owner or sysadmin or has select privilege of the sequence , then only the operation can be performed.

----End

4.10.7 Grant/Revoke Privilege

Follow the steps to grant/revoke privilege:

Step 1 Right-click schema group and select Grant/Revoke. The Grant/Revoke dialog is displayed.

Step 2 Select the objects to grant/revoke privilege from Object Selection tab and click Next.
Step 3  Select the role from Role drop-down in Privilege Selection tab.

Step 4  Select Grant/Revoke in Privilege Selection tab.

Step 5  Select/unselect the required privileges in Privilege Selection tab.

In SQL Preview tab, you can view the SQL query automatically generated for the inputs provided.

Step 6  Click Finish.

----End

4.10.8 Dropping a Schema

Individual or batch drop can be performed on schemas. Refer to Dropping Batch of Objects section for batch drop.

Follow the steps to drop a schema:

Step 1  In the Object Browser pane, right-click the selected schema and select Drop Schema.

A confirmation dialog to drop the schema is displayed.

Step 2  Click OK to drop the schema. This action will remove the schema from the Object Browser.

A popup message and status bar displays the status of the completed operation.

----End

4.11 Creating Function/Procedure

Follow the steps to create a function/procedure and SQL function:

Step 1  In the Object Browser pane, right-click Functions/Procedures under the particular schema where you want to create the function/procedure and select either Create PL/SQL Function or Create SQL Function or Create PL/SQL Procedure or Create C Function based on your requirement.

Data Studio opens a new tab with the selected template.

Step 2  Add the function/procedure by right-clicking in the tab and selecting Compile, or choosing Run > Compile/Execute Statement from the main menu, or pressing Ctrl+Enter to compile the function/procedure.

The Created function/procedure Successfully dialog box is displayed, and the new function/procedure is displayed under the Object Browser. Click OK to close the NewObject tab and add the debug object in the Object Browser.

Refer to the Execute SQL Queries section for information on the reconnect option in case connection is lost during execution.

Step 3  The * symbol next to the function/procedure name indicates that the function/procedure is not compiled or added in the Object Browser.

You must refresh (using F5) the Object Browser to view the newly added debug object.
NOTE

- A popup message displays the status of the completed operation. The status bar does not display the status of this operation.
- For C functions, debug operation is not supported.

----End

Supporting Compile Function

When a new PL/SQL Object, either from the template or by editing an existing PL/SQL Object, the newly created SQL Object will open in a new tab.

This new function will open in a new tab.

Follow the steps to compile function:

Step 1  Select Function/Procedures on object browser.

Step 2  Right-click on Function/Procedures. A menu option is displayed.

Step 3  Click Create PL/SQL Function. The new function/procedure tab is opened.
Step 4  Edit the code.

Step 5  Right-click on the tab. A menu option is displayed.

Step 6  Click Compile. A pop-up message is displayed as follows:

This function is displayed in a new tab.

4.12 Editing a Function/Procedure

Follow the steps to open and edit the function/procedure or SQL function:
Step 1 In the **Object Browser** pane, double-click the required function/procedure or SQL function or right-click the function/procedure or SQL function and select **View Source**. You must refresh the **Object Browser** to view the latest DDL.

The function/procedure or SQL function based on your selection is displayed.

```sql
CREATE OR REPLACE FUNCTION public.abcdef()
RETURNS integer
LANGUAGE sql
AS $$
  select 43544 as result
$$
```

Only one function/procedure or SQL function with the same schema, name, and input parameters can be opened in Data Studio.

Step 2 After editing or updating, compile and execute the PL/SQL program or SQL function. For more details, refer to **Executing a Function/Procedure**.

If you execute the function/procedure or SQL function before compiling, a **Source Code Change** dialog box is displayed.

Step 3 Click **Yes** to compile and execute the function/procedure.

The **Messages** tab displays the status of the completed operation.

Refer to **Execute SQL Queries** section for information on reconnect option in case connection is lost during execution.

Step 4 After compiling the function/procedure or SQL function, refresh the **Object Browser** (using F5) to view the updated code.

----End

### 4.13 Grant/Revoke Privilege

Follow the steps to grant/revoke privilege:

Step 1 Right-click functions/procedures group and select **Grant/Revoke**.

The **Grant/Revoke** dialog is displayed.

Step 2 Select the objects to grant/revoke privilege from **Object Selection** tab and click **Next**.

The **Privilege Selection** tab is displayed.

Step 3 Select the role from **Role** drop-down.

Step 4 Select **Grant/Revoke**.

Step 5 Select/unselect the required privileges.

The **SQL Preview** tab displays the SQL query automatically generated for the inputs provided.
4.14 Debugging a PL/SQL Function

4.14.1 Overview

During debugging operation if the connection is lost and the database is still connected in Object Browser, then Connection Error dialog box is displayed:

- **Yes** - The connection is reestablished and restarts debug operation.
- **No** - Disconnects database in Object Browser.

**NOTE**

SQL language function does not support debugging operations.

4.14.2 Using Breakpoints

This section contains the following topics:

- **Using the Breakpoints Pane**
- **Setting or Adding Breakpoints on a Line**
- **Enabling or Disabling a Breakpoint on a Line**
- **Removing a Breakpoint on a Line**
- **Source Code Change**
- **How to debug a PL/SQL program using breakpoints?**

A breakpoint is used to suspend the execution of a PL/SQL program at the line where the breakpoint is set. You can use breakpoints to control the execution and debug the function.

- An enabled breakpoint suspends the execution of the PL/SQL program whenever a breakpoint is encountered. When the execution hits the line of breakpoint, the execution will stop and you will be able to carry out other debug operations. Data Studio supports the following breakpoint operations:
  - Setting or Adding breakpoint on a line
  - Enabling or Disabling a breakpoint on a line
  - Removing a breakpoint on a line
- A disabled breakpoint will not suspend execution of PL/SQL program.

When you run a PL/SQL program, the execution pauses at every line where you set a breakpoint. When the program execution is paused, Data Studio retrieves information about the current program state, such as the values of the program variables.

Follow the steps to debug a PL/SQL program:

**Step 1** Set a breakpoint at the line where PL/SQL program execution should pause.

**Step 2** Start the debugging session.
When a line with a breakpoint is reached, monitor the state of the application in the debugger pane, and continue the execution.

**Step 3** Close the debugging session.

---End

Data Studio provides debugging options in the toolbar that helps you step through the debug objects.

**Using the Breakpoints Pane**

You can use the **Breakpoints** pane to view and manage the currently set breakpoints. From the minimized window panel, click the breakpoints option to open the breakpoints pane.

The **Breakpoints** pane lists each breakpoint with the line number and the debug object name.

You can enable or disable all the breakpoints by clicking □ in the **Breakpoints** pane. You can enable, disable or delete a specific breakpoint (s) by selecting the breakpoint check box and clicking ●, ○ or ✗ in the **Breakpoints** pane.

Double-click the required breakpoint in the **Breakpoint Info** column to locate the breakpoint in the **PL/SQL Viewer** pane.

![Breakpoints Pane](image)

<table>
<thead>
<tr>
<th>Line</th>
<th>Breakpoint Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>auto11() - integer</td>
</tr>
<tr>
<td>12</td>
<td>auto11() - integer</td>
</tr>
</tbody>
</table>

**NOTE**

- Disabling a breakpoint prevents the execution from pausing at the breakpoint, but leaves the definition in place (to enable the breakpoint later).
- Deleting a breakpoint removes it permanently.
- The content of the **Breakpoints** pane can be copied to the clipboard using **Alt+Y**.
Setting or Adding Breakpoints on a Line

Follow the steps to set or add breakpoints on a line:

**Step 1** Open the PL/SQL function where you want to add the breakpoint.

**Step 2** In the PL/SQL Viewer, double-click the breakpoint ruler on the left side of the line number column. The added breakpoint is indicated by an enable breakpoint sign ([ ] in the PL/SQL Viewer.

[NOTE] If the execution of the function does not break or stop during debugging, the breakpoint that is already set will not be validated.

----End

Enabling or Disabling a Breakpoint on a Line

Once a breakpoint is set, you can temporarily disable it by selecting the corresponding check box in the left-side of the Breakpoints pane and clicking [ ] at the top of the Breakpoints pane. Disabled breakpoints will be grayed out ([ ]) in the PL/SQL Viewer and Breakpoints pane.

To enable a disabled breakpoint, select the corresponding breakpoint (using check box) and click [ ].

Removing a Breakpoint on a Line

If you no longer require the breakpoint, you can remove it using the same actions used to create it.

In the PL/SQL Viewer tab, open the function in which you want to remove the breakpoint. Double-click the enable breakpoint sign ([ ]) in the PL/SQL Viewer to disable the breakpoint. The breakpoint is removed from the workspace.

You can also enable or disable breakpoints using the Breakpoints pane as explained above.

Source Code Change

During debugging, if the source code is changed after it is fetched from the server and debug is continued, Data Studio displays an error.

It is recommended that refresh the object and perform the debug operation again.

[NOTE] If the source code is changed after it is fetched from the server, and if you perform the execution or debug operation with no breakpoint set, then the result of the source code at the server will be displayed by Data Studio. It is always recommended to refresh before performing debug or execute operation.
How to debug a PL/SQL program using breakpoints?

Follow the steps to debug a PL/SQL program using breakpoints:

**Step 1** Open the PL/SQL program and add a breakpoint at the line where you want to perform debug operation.

For example:

Line 11, 12, 13.

**Step 2** To start debugging, click 🌟 or press Ctrl+D, or right-click the selected PL/SQL program in the **Object Browser** and select **Debug**. The **Debug Function/Procedure** dialog box appears prompting for your input.

**NOTE**

If there is no input parameter, then **Debug Function/Procedure** dialog box will not appear.

**Step 3** Provide your input and click **OK**. For varchar and date datatype, provide the input value in single quotes and for numeric datatype, provide the input value with or without single quotes.

To set NULL as the parameter value, enter **NULL** or **null**.

On clicking the **Debug** button, you will see an arrow 🌟 pointing to the line where breakpoint is set. The arrow indicates the line number at which execution will resume.
You can terminate debugging by clicking from the toolbar, or pressing F10, or select Terminate Debugging from the Debug menu. After stopping the debug operation, the execution of the function will not break for any breakpoint and the execution will proceed normally.

The Callstack and Variables panes are populated.

The Variables pane shows the current value of variables. Mouse over the variable in the function/procedure also shows the current value of variables.
You can step through the code using Step Into, Step Out or Step Over. For more details, refer to Controlling Execution.

Step 4 Click Continue to continue the execution till the next breakpoint (if any). The result of the executed PL/SQL program is displayed in the Result tab and the Callstack and Variables panes are cleared. You can copy the content of the Result tab, by clicking .

To remove the breakpoint, do the following:

- Double-click again on the breakpoint to remove it from the PL/SQL Viewer.
- Select the breakpoint in the breakpoint check box and click in the Breakpoints pane.

---End
Support Rearrangement Of Variable Window

This feature enables the Variable Window and columns to be rearranged. You are able to arrange Variable Window to the following places:

- Beside SQL Assistant Tab
- Beside SQL Terminal Tab
- Beside Object Browser Tab
- Beside Resultset Tab
- Beside Breakpoints Tab
- Beside Callstack Tab
- Below Object Browser Tab

**NOTE**

When debugging is finished, the variable window will be minimized even if the variable window is rearranged while debugging. If variable window is rearranged as a Terminal Tab or Result Tab, on completion of debugging, the tab should be minimized manually. The position of variable window is maintained after it is rearranged.

Enable/Disable System Variables

System Variables are displayed by default. You can disable the system variables whenever required.

**Step 1** Click the red button under Variables to disable System variables

![Variable Window](image)

The button is ON state by default.

---- End

Displaying Cached Parameter

During debug/execute a PL/SQL function or procedure, same parameter values are used for next time usage.

While executing PL/SQL object, following window is displayed.
For the first time, parameter values are empty. Enter the value as required.

![Parameter Values](image1.png)

and click OK. The parameter values will be cached. Next time during the query execution/debug same parameter values will be displayed.

**NOTE**

Once the specific connection is removed, all the parameter values in cache are cleared.

### Displaying Variable in Monitor Window

Data Studio displays the variables which are being monitored in the Monitor Window while performing debug operation.

In the Monitor Window variables must be added in following ways:

- Adding selected variables from the Variable Window and right click .
- Selecting the variables from variable window and add by clicking the button in the variable window toolbar.
If value is changed in the variable window, the same would defect in the monitor window if the variable is monitored and vice versa.

- Adding the variable by right click from the editor while debugging Function/Procedure.
The Monitor window can be dragged anywhere in the Data Studio window.

**Displaying Cursor Information For Variable During Debug**

In Data Studio, variable information is displayed if the cursor is hovered over that variable during debugging of PL/SQL function.

**Supporting Rollback/Commit During Debug**

Data Studio provides the option to commit/rollback the PL/SQL query execution result after debug is finished.

- If **Debug With Rollback** option is enabled, then after PL/SQL execution result after debug is not saved in the database.
- If **Debug With Rollback** option is disabled, then after PL/SQL execution result after debug is commited in the database.

Follow the steps to enable the rollback feature the feature:

**Step 1** Check the **Debug With Rollback** box to enable the rollback feature during PL/SQL debug.

Or

Right-click on SQL Terminal window where is PL/SQL function is executed.
Select **Debug With Rollback** option to enable rollback feature after debug is finished.

Or

Right-click on any PL/SQL function under **Functions/Procedure** in Object Browser.
4.14.3 Controlling Execution

This section contains the following topics:

- Starting the Debug Process
- Stepping through a PL/SQL Function
- Continuing the Debug Execution
- Viewing Callstack

Starting the Debug Process

Select the function that you want to debug in the Object Browser pane. Start debugging by clicking the button on the toolbar (or any other method as mentioned in the earlier sections). If no breakpoint is set, or the set breakpoint is invalid, debug operation will not halt at any statement and simply execute the object and display the results (if any).

Stepping through a PL/SQL Function

You can step through the debugging execution using the debug step commands from the toolbar. Step controls are used to step through the execution of the program line by line. If a breakpoint is encountered while performing a step operation, the execution will suspend at the breakpoint and the step operation is ended.

Stepping is the process of running one statement at a time. After stepping through a statement, you can see its effect in the other debugger tabs.
A maximum of 100 PL/SQL Viewer tabs can be displayed at a time. If a new tab beyond 100 is opened, the tab of the calling function is closed. For example, if 100 tabs are already opened and if one of the debug object calls a new debug object (other than already opened 100 tabs), then Data Studio will close the calling function, and open the new debug object.

**Step Into**

To step through code one statement at a time, select **Step Into** from the **Debug** menu, or press F7 button, or press F7.

When stepping into a function, Data Studio executes the current statement and then enters the break mode. The debug position will be indicated by an arrow on the left ruler pane. If the executed statement calls another function, Data Studio will step into that function. Once you have stepped through all the statements in that function, Data Studio will jump back to the next statement of the function it was called from.

To go into the next statement, press the **Step Into** (F7) button again. If you press the **Continue** button, PL/SQL code execution will continue as normal.

For example:

In the following example, when you step into Line 10, you will move to line 'm := F3_TEST();', that is, Line 9 in f3_test(). You can step through all the statements in f3_test() by stepping into each line by pressing the **Step Into** (F7) button repeatedly. Once you have stepped through all the statements in that function, Data Studio jumps to Line 10 in f2_test().

The currently debugging object is marked with symbol in the tab title with the function name.

**Step Over**

Step over is the same as step into, except that when it reaches a call for another function, it will not step into the function. The function will run, and you will be brought to the next statement in the current function. F8 is the shortcut key for step over. However, if there is a breakpoint set inside the called function, step over will enter the function, and hit the set breakpoint.

In the below example, when you click **Step Over** in Line 10, Data Studio runs the f3_test() function.
The cursor will be moved to the next statement in f2_test(), that is, Line 11 in f2_test().

You can step over a function when you are familiar with the way the function works and are sure that its execution will not affect the issue that you are investigating.

**NOTE**

Stepping over a line of code that does not contain a function call executes the line just like stepping into the line.

**Step Out**

Stepping out of a sub-program continues execution of the function and then suspends execution after the function returns to its calling function. You can step out of a long function when you have determined that the rest of the function is not significant to debug. However, if a breakpoint is set in the remaining part of the function, then that breakpoint will be hit before returning to the calling function.

Both stepping over and stepping out of a function will execute a function. The shortcut key for the step out operation is **Shift+F7**.

In the preceding example,
Choose **Debug > Step Into** to step into `f3_test()`.

Choose **Debug > Step Out** to step out of `f3_test()`.

**Continuing the Debug Execution**

When the debugging process stops at a specific location, you can select **Continue** (F9) from the **Debug Menu** or click on button from the toolbar to continue the PL/SQL function execution.

**Viewing Callstack**

The **Callstack** pane displays the chain of functions as they are called. The Callstack pane can be opened from the minimized window panel. The most recent functions are listed on the top, and the least recent on the bottom. At the end of each function name is the current line number in that function.

You can navigate among multiple functions through the **Callstack** pane by double-clicking the function name in the **Callstack** pane.

For example, when `f2_test()` calls `f3_test()` at Line 10, the debug pointer will point to the first valid executable line (which is Line 9, in the above example) in the called function.

In this case, the **Callstack** pane will be as shown below:

```
<table>
<thead>
<tr>
<th>Stack Info</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>f3_test()</code> - integer [Line : 9]</td>
</tr>
<tr>
<td><code>f2_test()</code> - integer [Line : 10]</td>
</tr>
</tbody>
</table>
```

**NOTE**

The content of the **Callstack** pane can be copied to the clipboard using Alt+J.

**4.14.4 Checking Debug Information**

When you use Data Studio, you can examine debugging information through several debug tabs. This section describes the operations that can be performed to check the debug information:

- **Operating on Variables**
- **Viewing Results**

**Operating on Variables**

The **Variables** pane is used to monitor information or evaluate values. The Variables pane can be opened from the minimized window panel. Using this pane, you can evaluate or modify variables or arguments in a PL/SQL function. As you step through the code, the values of some local variables may change.
The content of the **Variables** pane can be copied to the clipboard using `Alt+K`.

You can double-click the corresponding row of the variable and manually change variable values during run-time.

Click the **Variable**, **Datatype**, or **Value** column in the **Variables** pane to sort the values.

For example, to change the value of the percentage variable from 5 to 15, double-click the corresponding row in the **Variable** pane. The **Set Variable Value** dialog box will open, which prompts you to input the variable value. Provide the variable value and click **OK**.

To set NULL as the variable value, do not enter any value in text box.

If the variable is read-only, it will be indicated by 🛑 beside the corresponding variable.

A variable declared as a constant will not be shown as read-only in the **Variables** pane; however while updating it, an error will be seen.

![Variables pane](image)
In the Variables pane, the parameter value will be displayed as NULL, if the input to the parameter value is string literal 'NULL'.

When the value is set to a variable using Data Studio, then the value of the variable is same as the value returned by the statement "select expression' executed from gsql.

<table>
<thead>
<tr>
<th>Setting/Displaying Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting NULL Values</td>
<td>1. Double click on a variable value in Variables pane. A dialog box is displayed. 2. Make the value empty.</td>
</tr>
<tr>
<td>Setting String Values</td>
<td>• To set abc, enter abc. • To set string as Master's Degree enter Master&quot;s Degree. • To set variable as text(NULL), set NULL in Variables pane.</td>
</tr>
<tr>
<td>Setting Boolean Values</td>
<td>Enclose the boolean values t or f within single quotes. To set t to a boolean variable, enter 't' in the Variables pane.</td>
</tr>
<tr>
<td>Displaying Variable Value</td>
<td>If the variable value is NULL text, it will be displayed as NULL. If the variable value is NULL, it will be displayed as empty. If the variable value is a string, for example, abc, it will be displayed as abc.</td>
</tr>
</tbody>
</table>

Viewing Results

The Result tab displays the output for the PL/SQL debugging session, with the corresponding function/procedure name at the top of the tab. The Result tab will appear automatically, only if there is a result for the executed PL/SQL program.

You can copy the content of the Result tab, by clicking . Refer to Working with the SQL Terminals for more information.

NOTE

• The tool tip in the Result tab displays maximum of 10 lines, where each line contains maximum of 80 characters.
• If the result of the executed query is NULL, it will be displayed as <NULL>.
• Tab characters (ASCII 009) in table data will not be displayed in the Results/View Table Data/Properties window. Tab characters will be included correctly when copying/exporting the data. Tool tip will also display the tab characters correctly.
4.15 Working with Functions/Procedures

4.15.1 Overview

This section provides you with details on working with functions/procedures and SQL functions in Data Studio.

**NOTE**

Data Studio supports plpgsql and SQL languages for the operations are listed as follows:

- Creating Function/Procedure
- Editing a Function/Procedure
- Exporting a Function/Procedure DDL
- Dropping a Function/Procedure

4.15.2 Selecting a DB Object in the PL/SQL Viewer

Data Studio suggests a list of possible schema names, table names, column names, views, sequences, and functions in the PL/SQL Viewer.

Follow the steps below to select a DB object:

**Step 1** Press Ctrl+Space and enter the required parent DB object name. The DB objects list is refined as you continue typing the DB object name. The DB objects list displays all DB objects of the database connected to the SQL Terminal.

**Step 2** To select the parent DB object, use the Up or Down arrow keys and press Enter on the keyboard, or double-click the parent DB object.

**Step 3** Enter . (period) to list all child DB objects.
Step 4  To select the child DB object, use the Up or Down arrow keys and press Enter on the keyboard, or double-click the child DB object.

On selection, the child DB object will be appended to the parent DB object (with a period ':').

- Auto-suggest also works on schema names, table names, views, and table name aliases in the same way as shown above for all schema objects that you have access. Auto-suggest for column name alias is not supported.

Following is a sample query with alias objects:

```sql
SELECT
    table_alias.<auto-suggest>
FROM test.t1 AS table_alias
WHERE
    table_alias.<auto-suggest> = 5
GROUP BY table_alias.<auto-suggest>
HAVING table_alias.<auto-suggest> = 5
ORDER BY table alias.<auto-suggest>
```

- Auto-suggest may show "Loading" in Terminal for following scenarios:
  - The object is not loaded due to the value mentioned in the Load Limit field. Refer to Adding a Connection for more information.
  - The object is not loaded since it is added in the Exclude list option.
  - There is a delay in fetching the object from the server.

- If there are objects with the same name in different case, then auto-suggest will display child objects of both parent objects.

  **Example:**

  If there are two schemas with the name public and PUBLIC, then all child objects for both these schemas will be displayed.

--- End

4.15.3 Exporting a Function/Procedure DDL

Follow the steps below to export the Function/Procedure DDL:

Step 1  In the Object Browser pane, right-click the selected function/procedure and select Export DDL.

The Data Studio Security Disclaimer dialog box is displayed.

Step 2  Click OK.

The Save As dialog box is displayed.

Step 3  In the Save As dialog box, select the location to save the DDL and click Save. The status bar displays the progress of the operation.
- To cancel the export operation, double-click the status to open the Progress View tab and click ✗.
- The exported file name will not be the same as function/procedure name, if the function/procedure name contains characters which are not supported by Windows.
- Microsoft Visual C runtime file (msvcr100.dll) is required to complete this operation. Refer to Troubleshooting section for more information.
- Multiple objects can be selected to export DDL. Refer to Batch Export section for list of objects not supported for export DDL operation.

The Export message and status bar displays the status of the completed operation.

<table>
<thead>
<tr>
<th>Database Encoding</th>
<th>File Encoding</th>
<th>Supports Exporting DDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTF-8</td>
<td>UTF-8</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>GBK</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>LATIN1</td>
<td>Yes</td>
</tr>
<tr>
<td>GBK</td>
<td>GBK</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>UTF-8</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>LATIN1</td>
<td>No</td>
</tr>
<tr>
<td>LATIN1</td>
<td>LATIN1</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>GBK</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>UTF-8</td>
<td>Yes</td>
</tr>
</tbody>
</table>

4.15.4 Viewing Object Properties in the PL/SQL Viewer

Data Studio allows you to view table properties, procedures/functions and SQL functions.

Follow the steps below to view table properties:

**Step 1** Press Ctrl and point to the table name.
Step 2  Click the highlighted table name.

The properties of the selected table is displayed.

NOTE

The table properties are read-only.

----End

Follow the steps below to view functions/procedures or SQL functions:

Step 1 Press Ctrl and point to the procedure/function name or SQL function name.

Step 2 Click the highlighted function/procedure name or SQL function name. The function/procedure or SQL function is displayed in a new PL/SQL Viewer tab based on your selection.

----End

Follow the steps below to View Object DDL:

Step 1 Press Ctrl and point to the View Object DDL name.

Step 2 Click the highlighted View Object DDL name. The View Object DDL is displayed in a new tab based on your selection.

----End

4.15.5 Dropping a Function/Procedure

Individual or batch drop can be performed on functions/procedures. Refer to Dropping Batch of Objects section for batch drop.

Follow the steps below to drop a function/procedure or SQL function object:
Step 1 In the Object Browser pane, right-click the selected function/procedure object and select Drop Object.

Step 2 To drop objects in batches, right-click two or more selected function/procedure objects and choose Drop Objects.

Step 3 In the confirmation dialog box, click Yes to complete the operation successfully. The status bar displays the status of the completed operation.

----End

4.15.6 Executing a Function/Procedure

After you connect to the database, all the stored functions/procedures and tables will be automatically populated in the Object Browser pane. You can use Data Studio to execute PL/SQL programs or SQL functions.

**NOTE**

- Blank lines occurring above or below in a function/procedure will be trimmed by Data Studio before being sent to the server. Blank lines will also be trimmed when displaying the source received from the server.
- To execute a function/procedure, enter the same values in Data Studio and the gsql client. If you do not enter any value in Data Studio, then NULL is considered as the input.
  
  For example:
  - To execute the function/procedure with string, enter the value as `data`
  - To execute the function/procedure with date, enter the value as `to_date ('2012-10-10', 'YYYY-MM-DD');`
- A function/procedure with OUT and INOUT parameter types cannot be executed directly.
- Data Studio will not execute any function/procedure with unknown data type parameters.

You can right-click the function/procedure in the Object Browser to perform the following operations:

- Refresh the program to get the latest program from the server
- Execute the function/procedure or SQL function
- Debug the PL/SQL function
- Drop the debug object

How to execute a PL/SQL program or SQL function?

Follow the steps to execute a PL/SQL program or SQL function:

Step 1 Double-click and open the PL/SQL program or SQL function. Each debug object opens in a new tab. You can open a maximum of 100 tabs in Data Studio.

Step 2 Click 🍀 in the toolbar, or choose Run > Execute DB Object from the main menu, or right-click the program name in the Object Browser and select Execute.

Alternatively, you can right-click in the PL/SQL Viewer tab and select Execute.

Step 3 The Execute Function/Procedure dialog box is displayed prompting for your input.
### 4.15.7 Grant/Revoke Privilege

Follow the steps below to grant/revoke privilege:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Right-click selected function/procedure and select <strong>Grant/Revoke</strong>.</td>
</tr>
<tr>
<td></td>
<td>The <strong>Grant/Revoke</strong> dialog is displayed.</td>
</tr>
<tr>
<td>2</td>
<td>Refer to <strong>Grant/Revoke Privilege</strong> section to grant/revoke privilege.</td>
</tr>
</tbody>
</table>

----End

### 4.16 Tables

#### 4.16.1 Overview

This section describes how to work with tables effectively.
NOTE
You need to fill all the mandatory parameters that are marked with asterisk (*) to complete the operation successfully.

4.16.2 Creating Regular Table

4.16.2.1 Overview

This section describes the steps to create a Regular table.

Tables are logical structures maintained by the database manager. Tables are made up of columns and rows. You can define tables as part of your data definitions in the data perspective. Before you can define a table, you must first have a database and a schema defined.

This section shows you how to create new table using Data Studio.

Follow the steps to define a table in your database:

Step 1 In the Object Browser pane, right-click Regular Tables, and select Create Regular Table.

Step 2 Provide basic table information such as table name, tablespace, table type and so on. For more details, refer to Providing General Information. To create a tablespace, refer to Working with Tablespaces.

Step 3 Define column related information such as column name, data type schema, data type, and column constraints. For more details, refer to Defining Columns.

Step 4 Select the data distribution information for the table. For more details, refer to Selecting Data Distribution.

Step 5 Define the column constraints for different constraint types such as primary key, unique, and check. For more details, refer to Defining Table Constraints.

Step 6 Define the index information for the table such as index name, access method, tablespace and so on. For more details, refer to Defining Indexes.

In SQL Preview tab, you can view the SQL query automatically generated for the inputs provided. For more details, refer to SQL Preview.

---End

Providing General Information

When you create a table within a schema, the current schema is used as the table’s schema. There are several steps involved in creating a table.

Provide the following information to create a regular table:

Step 1 Enter the table name in the Table Name field. It specifies the name of the table to be created.
NOTE

Select Case check box to retain the capitalization of the text entered in Table Name field. For example, if the table name entered is “Employee”, then the table name is created as “Employee”.

The schema name under which the table is created is displayed in the Schema drop-down.

Step 2  Select table orientation from Table Orientation drop-down.

Step 3  Select the tablespace from Tablespace drop-down. It specifies the name of the tablespace in which the new table is to be created. If not provided, the default tablespace of the table’s schema will be used.

Step 4  Select the table type from Table Type drop-down. It specifies the type of the table.

- Normal: If specified, the table is created as a normal table.
- Unlogged: If specified, the table is created as an un-logged table. Data written to un-logged tables is not written to the write-ahead log, which makes it considerably faster than ordinary tables. However, it is not crash-safe. An un-logged table is automatically truncated after a crash or unclean shutdown. The contents of an un-logged table are also not replicated to standby servers. Any indexes created on an un-logged table are also automatically un-logged.

Step 5  Select the required Options.

- IF NOT EXISTS check box to create the table only if table with same name does not exist.
- WITH OIDS check box for the new table to have object identifiers (OID) assigned. If you need a new table with OIDs, choose this option.
- Select the Fill Factor. The fill factor for a table is a percentage between 10 and 100. 100 (complete packing) is the default value. When a smaller fill factor is specified, INSERT operations pack table pages only to the indicated percentage; the remaining space on each page is reserved for updating rows on that page. This gives UPDATE operation a chance to place the updated copy of a row on the same page as the original, which is more efficient than placing it on a different page. For a table whose entries are never updated, complete packing is the best choice, but in heavily updated tables, smaller fill factors are appropriate. This parameter cannot be set for TOAST tables.

Step 6  Enter the description of the table in Description of Table box. It specifies a short note on the table.

Step 7  After providing the general information about the table, click Next to define the columns information for the table.

----End

Following Table lists the supported fields for each Regular type tables:
Table 4-5 Supported Fields

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Row Table</th>
<th>Column Table</th>
<th>ORC Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tablespace</td>
<td>Normal</td>
<td>Normal</td>
<td>Hadoop Distributed File System (HDFS)</td>
</tr>
<tr>
<td>Table Type</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>If Not Exists</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>With OIDS</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Fill Factor</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

Defining Columns

A column defines a unit of information within a table’s row. Each row is an entry in the table. Each column is a category of information that applies to all rows. When you add a table to a database, you can define the columns that compose it. Columns determine the type of data that the table can hold.

After providing the general information about the table, click the **Columns** tab to define the list of table columns. Each column contains name, data type, and other optional properties.

You can perform the following operations on an existing column only for a Regular table:

- Deleting a Column
- Editing a Column
- Moving a Column

Follow the steps to define column(s) for the table:

**Step 1** Enter the column name in **Column Name** field. It specifies the name of a column to be created in the new table. This must be a unique name in the table.

**NOTE**

Select **Case** check box to retain the capitalization of the text entered in **Column Name** field. For example, if the column name entered is “Name", then the column name is created as “Name”.

**Step 2** Select the **Array Dimensions**. It specifies the array dimensions for the column.

**Example:** If array dimension for a column is defined as integer [ ], then it will add the column data as single dimension array.
The **marks** column in the above table was created as single dimension and **subject** column as two dimensions.

**Step 3** Select the data type of the column from **Data Type** drop-down. For example, **bigint** for integer values.

For complex data types,
- Select the required schema from the **Data type Schema** drop-down list.
- Select the corresponding data type from the **Data Type** drop-down list. This list displays the tables and views for the selected schema.

**NOTE**

User defined data type will not be available for selection.

**Step 4** Enter the precision/size value of the datatype entered in the **Precision/Size** field. This option is available only if a data type can be defined with precision/size.

**Step 5** Select the scale of the data type entered in the **Scale** field.

**Step 6** Choose the following **Column Constraints** if required:
- **NOT NULL** - Specifies that this column is not allowed to contain null values.
- **UNIQUE** - Specifies that a column may contain only unique values.
- **DEFAULT** - Specifies the value that will be used for this column in case no value is defined.
- **CHECK** - Specifies an expression producing a boolean result which new or updated rows must satisfy for an INSERT or UPDATE operation to succeed.

**Step 7** After you enter all information for new column, click **Add**. You can also delete a column from a list or change the order of columns. After defining all columns, click **Next**.

---End

Following table lists the supported fields for each Regular type tables:

**Table 4-6 Supported Fields**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Row Table</th>
<th>Column Table</th>
<th>ORC Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array Dimensions</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Data type Schema</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Not Null</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
Deleting a Column

Follow the steps to delete a column:

**Step 1** Select the required column.

**Step 2** Click **Delete**.

----End

Editing a Column

Follow the steps to edit a column:

**Step 1** Select the required column.

**Step 2** Click **Edit**.

**Step 3** Edit the column details as required and click **Update** to save changes.

**NOTE**

You must complete the edit operation and save the changes to continue with other operations.

----End

Moving a Column

You can move a column to change the location of the column as required in the table. To move a column, select the required column and click **Up** or **Down**.

Selecting Data Distribution

Data distribution specifies how the table is distributed or replicated among data nodes.

Select one of the following options for the distribution type:

<table>
<thead>
<tr>
<th>Distribution Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT DISTRIBUTION</td>
<td>Specifies that the default distribution type will be assigned for this table.</td>
</tr>
</tbody>
</table>
### Distribution Type

<table>
<thead>
<tr>
<th>Distribution Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPLICATION</td>
<td>Each row of the table will be replicated in all the data nodes of the database cluster.</td>
</tr>
<tr>
<td>HASH</td>
<td>Each row of the table will be placed based on the hash value of the specified column.</td>
</tr>
<tr>
<td>RANGE</td>
<td>Each row of the table will be placed based on the range value.</td>
</tr>
<tr>
<td>LIST</td>
<td>Each row of the table will be placed based on the list value.</td>
</tr>
</tbody>
</table>

After selecting data distribution, click **Next**.

Table below lists the supported options for each Regular type tables:

**Table 4-7 Supported Options**

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Row Table</th>
<th>Column Table</th>
<th>ORC Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
</tr>
<tr>
<td>Hash</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Replication</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
</tr>
</tbody>
</table>

### Defining Table Constraints

Creating constraints is optional. A table can have one (and only one) primary key. Creating the primary key is a good practice.

You can select the following types of constraints from the **Constraint Type** drop-down list:

- **Primary Key**
- **Unique**
- **Check**

### Primary Key

The primary key is the unique identity of a row and consists of one or more columns.

Only one primary key can be specified for a table, either as a column constraint or as a table constraint. The primary key constraint must name a set of columns that is different from other sets of columns named by any unique constraint defined for the same table.
Select the constraint type as **PRIMARY KEY** in the combo box and enter the constraint name. Select the column from **Available Columns** list and click **Add**. If you need a multi-column primary key, repeat this step for another column.

Optionally, you can select **On Tablespace** in which the index associated with a **PRIMARY KEY** constraint will be created. If this parameter is not provided, the index will be created in the same tablespace as the table.

**Fillfactor** for a table is a percentage between 10 and 100. The default value is 100 (complete packing). When a smaller fill factor is specified, INSERT operations will pack table pages only up to the indicated percentage; the remaining space on each page is reserved for updating rows on that page. This gives UPDATE operation a chance to place the updated copy of a row on the same page as the original, which is more efficient than placing it on a different page.

For a table whose entries are never updated, complete packing is the best choice, but in heavily updated tables, smaller fill factors are appropriate. This parameter cannot be set for TOAST tables.

**DEFERRABLE**: Select this check box to defer this option.

**INITIALLY DEFERRED**: Select this check box to check the constraint at the set default time.

Click **Add** in the **Constraints** group box.

You can remove a primary key from the list using the **Delete** button.

Mandatory parameters are marked with asterisk (*) in the corresponding field.

---

**Unique**

Select the constraint type as **UNIQUE** in the combo box and enter the constraint name in the text box.

Select column in **Available Columns** list and click **Add**. If you need multi-column unique, repeat this step for another column. After adding the first column, the unique name is automatically filled from the table name. You can also change this name.

Optionally, you can select **Tablespace** in which the index associated with a unique constraint needs to be created. If this parameter is not provided, the index will be created in the same tablespace as the table.

- **Fillfactor**: Refer to **Primary Key** section for fill factor information.
- **DEFERRABLE**: Refer to **Primary Key** section for deferral information.
- **INITIALLY DEFERRED**: Refer to **Primary Key** section for initially deferred information.

You can a remove unique from the list using **Delete** button are marked with asterisk (*) in the corresponding field.

---

**Check**

Select the constraint type as **CHECK** in the combo box and enter the constraint name in the text box.
When the INSERT or UPDATE operation is performed, and if the check expression fails, then table data is not altered.

If you double-click on column in **Available Columns** list, it is inserted to **Check Expression** edit line to current cursor position.

Then, click **Add** in **Constraints** group box. You can also remove a check from the list using **Delete** button. Mandatory parameters are marked with asterisk (*) in the corresponding field.

After defining all constraints, click **Next**.

Following table lists the supported options for each Regular type tables:

**Table 4-8 Supported Options**

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Row Table</th>
<th>Column Table</th>
<th>ORC Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Unique</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Primary Key</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

**Defining Indexes**

Creating indexes is optional. Indexes are primarily used to enhance database performance. This operation constructs an index on the specified column(s) of the specified table. Select the **Unique Index** check box to enable this option.

Choose the name of the index method from the **Access Method** list. The default method is B-tree. Select the **Tablespace** in which the index must be created. If not specified, the index is created in the default tablespace.

The **Fillfactor** for an index is a percentage that determines how full the index method will try to pack index pages. For B-trees, leaf pages are filled to this percentage during initial index build, and also when extending the index at the right (adding new largest key values). If pages subsequently become completely full, they will be split, leading to gradual degradation in the index's efficiency.

B-trees use a default fill factor of 90, but any integer value from 10 to 100 can be selected. If the table is static, then a fill factor of 100 to minimize the index's physical size. For heavily updated tables, a explain plan smaller fill factor is better to minimize the need for page splits. Other index methods use fill factor in different but roughly analogous ways; the default fill factor varies between methods.

You can either enter a user-defined expression for the index or you can create the index using the **Available Columns** list. Select the column in the **Available Columns** list and click **Add**. If you need a multi-column index, repeat this step for other columns.

After entering the required information for the new index, click **Add**.
You can also delete an index from the list using the Delete button. After defining all indexes, click Next.

Following Table lists the supported fields/options for each Regular type tables:

### Table 4-9 Supported Fields/Options

<table>
<thead>
<tr>
<th>Field/Option Name</th>
<th>Row Table</th>
<th>Column Table</th>
<th>ORC Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique Index</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>btrees</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>gins</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>gist</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>hash</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>psorts</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>spgists</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Tablespace</td>
<td>Normal</td>
<td>Normal</td>
<td>✗</td>
</tr>
<tr>
<td>Fill Factor</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>User Defined Expression</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Partial Index</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

**SQL Preview**

Data Studio generates a DDL statement based on the inputs provided in Create New table wizard.

You can only view, select, and copy the query. You cannot edit the query.

- To select all queries, press Ctrl+A or right-click and select Select All.
- To copy the selected query, press Ctrl+C or right-click and select Copy.

Click Finish to create the table. On clicking the Finish button, the generated query will be sent to the server. Any errors are displayed in the dialog box and status bar.

### 4.16.2.2 Working with Columns

After creating a table, you can add new columns in that table. You can also perform the following operations on the existing column only for a Regular table:
Creating New Column

Follow the steps to add a new column to the existing table:

**Step 1** Right-click **Columns** and select **Create column**.

The **Add New Column** dialog box is displayed prompting you to add information about the new column.

**Step 2** Enter the details and click **Add**. You can view the added column in the corresponding table.

Data Studio displays the status of the operation in the status bar.

---End

Rename Column

Follow the steps to rename a column:

**Step 1** Right-click the selected column and select **Rename Column**.

A **Rename Column** dialog box is displayed prompting you to provide the new name.

**Step 2** Enter the name and click **OK**. Data Studio displays the status of the operation in the status bar.

---End

Toggle Not Null

Follow the steps to set or reset the Not Null option:

**Step 1** Right-click the selected column and select **Toggle Not Null**.

A **Toggle Not Null Property** dialog box is displayed prompting you to set or reset the Not Null option.

**Step 2** In the confirmation dialog box, click **OK** to complete the operation successfully. Data Studio displays the status of the operation in the status bar.

---End

Drop Column

Follow the steps to drop the column:

**Step 1** Right-click the selected column and select **Drop Column**. This operation deletes the column from the table.
A Drop Column dialog box is displayed.

**Step 2** Click OK to complete the operation successfully. Data Studio displays the status of the operation in the status bar.

----End

**Set Column Default**

Follow the steps to set the default value for the column:

**Step 1** Right-click the selected column and select Set Column Default Value.

A dialog box with the current default value (if it is set) and prompting you to provide the default value is displayed.

**Step 2** Enter the value and click OK. Data Studio displays the status of the operation in the status bar.

----End

**Change Data Type**

Follow the steps to change the data type of the column:

**Step 1** Right-click the selected column and select Change Data Type.

Change Data Type dialog box is displayed.

- **NOTE**

  The existing data type shows as Unknown while modifying complex data types.

**Step 2** Select the Data type Schema and Data Type. If the Precision/Size spin box is enabled, enter the required details and click OK. Data Studio displays the status of the operation in the status bar.

----End

**4.16.2.3 Working with Constraints**

You can perform the following operations after a table is created only for a Regular table:

- Creating a Constraint
- Renaming a Constraint
- Dropping a Constraint

**Creating a Constraint**

Follow the steps to add a new constraint to the existing table:

**Step 1** Right-click the selected constraint of the table and select Create constraint.

An Add New Constraint dialog box is displayed prompting you to add information about the new constraint.
Step 2 Enter the **Constraint Name**, **Check Expression**, and click **Add**. You can view the added constraint in the corresponding table.

Data Studio displays the status of the operation in the status bar.

---

**NOTE**

The status bar will show the name of the constraint if it has been provided in the **Constraint Name** field, else the constraint name will not be displayed as it is created by database server.

----End

**Renaming a Constraint**

Follow the steps to rename a constraint:

Step 1 Right-click the selected constraint and select **Rename Constraint**.

A **Rename Constraint** dialog box is displayed prompting you to provide the new name.

Step 2 Enter the constraint name and click **OK**. Data Studio displays the status of the operation in the status bar.

----End

**Dropping a Constraint**

Follow the steps to drop the constraint:

Step 1 Right-click the selected constraint and select **Drop Constraint**.

A **Drop Constraint** dialog box is displayed.

Step 2 Click **OK** to complete the operation successfully. Data Studio displays the status of the operation in the status bar.

----End

**4.16.2.4 Working with Indexes**

An index can be created in a table to find data quickly and efficiently.

After creating a table, you can add a new index to that table. You can also perform the following operations on an existing index only for a Regular table:

- Creating a New Index
- Renaming an Index
- Changing the Tablespace
- Changing the Fill Factor
- Dropping an Index

**Creating a New Index**

Follow the steps to add a new index to the existing table:
Step 1 Right-click Indexes and select Create index.

A Create Index dialog box is displayed prompting you to add information about the index.

Step 2 Enter the details and click Create. You can also view the SQL statement by clicking the Preview Query button. Items in Available Columns are not sorted. Items moved back from Index Columns to Available Columns are unsorted, and is not related to the column order in the table. You can set the order of the Index Columns using the arrow buttons. Data Studio displays the status of the operation in the status bar.

----End

Renaming an Index

Follow the steps to rename an index:

Step 1 Right-click the selected index and select Rename Index.

A Rename Index dialog box is displayed prompting you to provide the new name.

Step 2 Enter the name and click OK. Data Studio displays the status of the operation in the status bar.

----End

Changing the Tablespace

Follow the steps to change the tablespace:

Step 1 Right-click the selected index and select Change Tablespace.

A Change Tablespace dialog box is displayed prompting you to select the tablespace details.

Step 2 Select the tablespace and click OK. Data Studio displays the status of the operation in the status bar.

----End

Changing the Fill Factor

Follow the steps to change the fill factor:

Step 1 Right-click the selected index and select Change Fill Factor.

A Change Fill Factor dialog box is displayed prompting you to select the fill factor details.

Step 2 Select the fill factor and click OK. Data Studio displays the status of the operation in the status bar.

----End

Dropping an Index

Follow the steps to drop an index:
Step 1  Right-click the selected index and select **Drop Index**. Data Studio prompts you to confirm this operation.

The **Drop Index** dialog box is displayed.

Step 2  Click **OK** to complete the operation successfully. Data Studio displays the status of the operation in the status bar. This operation deletes the index from the table.

**NOTE**

When the last index of a table is dropped and if the table properties are checked, then **Has Index** may reflect the value "TRUE", though the table has no index. This value is updated to "FALSE" when a vacuum operation on the table is performed.

--- End

4.16.3 Creating Foreign Table

Foreign tables created using query execution in SQL Terminal or any other tool can be viewed in the Object browser after refresh.

Step 1  To view the newly created foreign table, right-click and select **Refresh** either at database, schema and foreign table group level.

**NOTE**

- GDS Foreign table is denoted with icon before the table name.
- HDFS Foreign table is denoted with icon before the table name.
- HDFS Foreign table with partition is denoted with icon before the table name.

--- End

4.16.4 Creating Partition Table

4.16.4.1 Overview

Partitioning refers to splitting what is logically one large table into smaller physical pieces based on specific schemes. The table based on the logic is called a partition table, and a physical piece is called a partition. Data is stored on these smaller physical pieces, namely, partitions, instead of the larger logical partition table.

Follow the steps to define a table in your database:

Step 1  In the **Object Browser** pane, right-click **Regular Tables**, and select **Create Partition Table**.

Step 2  Provide basic table information such as table name, tablespace, table type and so on. For more details, refer to **Providing General Information**.

Step 3  Define column related information such as column name, data type schema, data type, and column constraints. For more details, refer to **Defining Columns**.

Step 4  Select the data distribution information for the table. For more details, refer to **Selecting Data Distribution**.
Step 5  Define the column constraints for different constraint types such as primary key, unique, and check. For more details, refer to Defining Table Constraints.

Step 6  Define the index information for the table such as index name, access method, tablespace and so on. For more details, refer to Defining Indexes.

Step 7  Define the partition information for the table such as partition name, partition column, partition value and so on. For more details, refer to Defining Partitions.

In SQL Preview tab, you can view the SQL query automatically generated for the inputs provided. For more details, refer to SQL Preview.

---End

Providing General Information

Provide the following information to create a table:

For information on completing the below fields refer to Providing General Information.

- Table Name
- Schema
- Tablespace
- Options
- Description of Table

For completing all other fields refer as follows:

Step 1  Select table orientation from Table Orientation drop-down.

**NOTE**
If table orientation is selected as Optimized Row Columnar (ORC), then a HDFS Partition table is created.

Step 2  Enter the ORC version number in the ORC Version field. This is applicable only for HDFS Partition table.

Step 3  After providing the general information about the table, click Next to define the columns information for the table.

Table below lists the supported fields for each Partition type tables:

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Row Partition</th>
<th>Column Partition</th>
<th>ORC Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tablespace</td>
<td>Normal</td>
<td>Normal</td>
<td>HDFS</td>
</tr>
<tr>
<td>Table Type</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>If Not Exists</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>With OIDs</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
Defining Columns

Refer to Defining Columns to define column(s) for the table.

Table below lists the supported fields for each Partition type tables:

Table 4-11 Supported Fields

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Row Partition</th>
<th>Column Partition</th>
<th>ORC Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fill Factor</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

Change Order of Partition

You can change the order of partition as required in the table. To change the order, select the required partition and click Up or Down.

SQL Preview

Refer to SQL Preview

Editing a Partition

Follow the steps below to edit a partition:

Step 1  Select the required partition.

Step 2  Click Edit.

Step 3  Edit the partition details as required and click Update to save changes.
You must complete the edit operation and save the changes to continue with other operations.

----End

Deleting a Partition

Follow the steps below to delete a partition:

Step 1 Select the required partition.

Step 2 Click Delete.

----End

Defining Partitions

Table below lists the supported fields/options for each Partition type tables:

<table>
<thead>
<tr>
<th>Field/Option Name</th>
<th>Row Partition</th>
<th>Column Partition</th>
<th>ORC Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition Type</td>
<td>By Range</td>
<td>By Range</td>
<td>By Values</td>
</tr>
<tr>
<td>Partition Name</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Partition Value</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Tablespace</td>
<td>Normal</td>
<td>Normal</td>
<td>✗</td>
</tr>
</tbody>
</table>

Follow the steps below to define partition(s) for the table:

Step 1 If Row or Column is selected as Table Orientation in the General tab, then By Range is displayed in the Partition Type section. If ORC is selected as Table Orientation in the General tab, then By Value is displayed in the Partition Type section.

Step 2 Select the column based on which partition needs be defined from the Available Column section and click .

The column moves to the Partition Column section.
NOTE

- If Table Orientation is selected as Row or Column, then only one column can be selected for partition.
- If Table Orientation is selected as ORC, then maximum of four columns can be selected for partition.
- A maximum of 4 columns can be selected to define partition.

Step 3 Enter a name for the partition in Partition Name field.

Step 4 Click next to the Partition Value field.
   1. Enter the value by which you want to partition the table in Value column.
   2. Click OK.

Step 5 Select the tablespace name from the Tablespace drop-down.

NOTE

Partition Name, Partition Value, Tablespace and Partitions section are disabled for ORC Partition tables.

Step 6 After you enter all information for partition, click Add.

Step 7 After defining all partitions, click Next.

----End

You can perform the following operations on an existing partition for Row or Column Partition table. Below operations are not applicable for ORC Partition table:

- Deleting a Partition
- Editing a Partition

Defining Indexes

Refer to Defining Indexes to define table indexes.

Table 4-13 Supported Options

<table>
<thead>
<tr>
<th>Field/Option Name</th>
<th>Row Partition</th>
<th>Column Partition</th>
<th>ORC Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique Index</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>btree</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>gin</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>gist</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>hash</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Field/Option Name</td>
<td>Row Partition</td>
<td>Column Partition</td>
<td>ORC Partition</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------</td>
<td>------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>psort</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>spgist</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Tablespace</td>
<td>Normal</td>
<td>Normal</td>
<td>✓</td>
</tr>
<tr>
<td>Fill Factor</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>User Defined Expression</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Partial Index</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Defining Table Constraints**

Refer to **Defining Table Constraints** to define table constraint(s).

**Table 4-14 Supported Options**

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Row Partition</th>
<th>Column Partition</th>
<th>ORC Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Unique</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Primary Key</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Selecting Data Distribution**

Refer to **Selecting Data Distribution** for selecting the distribution type.

**Table 4-15 Supported Options**

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Row Partition</th>
<th>Column Partition</th>
<th>ORC Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Hash</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Replication</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
4.16.4.2 Working with Partitions

After creating a table, you can add/modify partitions. You can also perform the following operations on an existing partition:

- **Rename a Partition**
- **Drop a Partition**

**Rename a Partition**

Follow the steps below to rename a partition:

**Step 1** Right-click the selected partition and select **Rename Partition**.

**Rename Partition Table** dialog box is displayed prompting you to provide the new name for the partition.

**Step 2** Enter new name and click **OK**.

Data Studio displays the status of the operation in the status bar.

----End

**Drop a Partition**

Follow the steps below to drop a partition:

**Step 1** Right-click the selected index and select **Drop Partition**.

**Drop Partition Table** dialog box is displayed.

**Step 2** Click **OK**.

The partition is dropped from the table. Data Studio displays the status of the operation in the status bar.

----End

4.16.5 Grant/Revoke Privilege - Regular/Partition Table

Follow the steps below to grant/revoke privilege:

**Step 1** Right-click regular tables group and select **Grant/Revoke**.

The **Grant/Revoke** dialog is displayed.

**Step 2** Select the objects to grant/revoke privilege from **Object Selection** tab and click **Next**.

**Step 3** Select the role from **Role** drop-down in **Privilege Selection** tab.

**Step 4** Select **Grant/Revoke** in **Privilege Selection** tab.

**Step 5** Select/unselect the required privileges in **Privilege Selection** tab.

In **SQL Preview** tab, you can view the SQL query automatically generated for the inputs provided.
Step 6  Click Finish.

----End

4.16.6 Managing Table

4.16.6.1 Overview

This section describes how to manage tables effectively.

NOTE

- You need to fill all the mandatory parameters, that are marked with asterisk (*) to complete the operation successfully.
- Refresh is the only operation supported for foreign table.

After creating the table, you can perform operations on the existing table. Right-click the selected table and select the required operation.

Context Menu

Additional options for table operations are available in the table context menu.

The context menu options available for table operations are:

<table>
<thead>
<tr>
<th>Table 4-16 Table Context Menu Options</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Option</strong></td>
</tr>
<tr>
<td>View Table Data</td>
</tr>
<tr>
<td>Edit Table Data</td>
</tr>
<tr>
<td>Re-index Table</td>
</tr>
<tr>
<td>Analyze Table</td>
</tr>
<tr>
<td>Truncate Table</td>
</tr>
<tr>
<td>Vacuum Table</td>
</tr>
<tr>
<td>Set table Description</td>
</tr>
<tr>
<td>Set Tablespace</td>
</tr>
<tr>
<td>Set Schema</td>
</tr>
<tr>
<td>Export Table Data</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>Import Table Data</td>
</tr>
<tr>
<td>Show DDL</td>
</tr>
<tr>
<td>Export DDL</td>
</tr>
<tr>
<td>Export DDL and Data</td>
</tr>
<tr>
<td>Rename Table</td>
</tr>
<tr>
<td>Drop Table</td>
</tr>
<tr>
<td>Properties</td>
</tr>
<tr>
<td>Grant/Revoke</td>
</tr>
<tr>
<td>Refresh</td>
</tr>
</tbody>
</table>

### 4.16.6.2 Renaming a Table

Follow the steps below to rename the table:

**Step 1** Right-click the selected table and select **Rename Table**.  
A **Rename Table** dialog box is displayed prompting you to provide the new name.

**Step 2** Enter the table name and click **OK**. You can view the updated table name in the **Object Browser**.  
Data Studio displays the status of the operation in the status bar.

**NOTE**  
This operation is not supported for Partition ORC tables.

----End

### 4.16.6.3 Truncating a Table

Follow the steps below to truncate the table:

**Step 1** Right-click the selected table and select **Truncate Table**. This operation deletes the data from an existing table.  
Data Studio prompts you to confirm this operation.

**Step 2** In the confirmation dialog box, click **OK** to complete the operation successfully.  
A popup message and status bar displays the status of the completed operation.

----End
4.16.6.4 Reindexing a Table

Index helps with faster lookup of records. You need to re-index tables in the following scenarios:

- An index is corrupted and no longer contains valid data. Although in theory this must never happen, in practice, indexes can become corrupted due to software bugs or hardware failures. Re-index provides a recovery method.
- An index has become "bloated", that is, it contains many empty or nearly-empty pages. This can occur with B-tree indexes in PostgreSQL under certain uncommon access patterns. Reindex provides a way to reduce the space consumption of the index by writing a new version of the index without the dead pages.
- You have altered a storage parameter (such as fill factor) for an index, and wish to ensure that the change has taken full effect.

Follow the steps to re-index a table:

**Step 1** Right-click the selected table and select **Reindex Table**.

A popup message and status bar displays the status of the completed operation.

**NOTE**

This operation is not supported for Partition ORC tables.

----End

4.16.6.5 Analyzing a Table

The analyze table operation gathers statistics about tables and indices of that table and stores the collected information in internal tables of the database where the query optimizer can access the information and use it to help make better query planning choices.

Follow the steps to analyze a table:

**Step 1** Right-click the selected table and select **Analyze Table**.

The **Analyze Table** message and status bar displays the status of the completed operation.

----End

4.16.6.6 Vacuuming a Table

Vacuum table operation reclaims space and makes it available for re-use.

Follow the steps to vacuum the table:

**Step 1** Right-click the selected table and select **Vacuum Table**.

The **Vacuum Table** message and status bar displays the status of the completed operation.

----End
4.16.6.7 Setting the Table Description

Follow the steps to set the description of the table:

**Step 1** Right-click the selected table and select **Set Table Description**.

An **Update Table Description** dialog box is displayed. It prompts you to provide the table description.

**Step 2** Enter the description and click **OK**.

The status bar displays the status of the completed operation.

**Step 3** To view the table description, right-click selected the table and select **Properties**.

---End

4.16.6.8 Setting the Tablespace

Follow the steps to set the tablespace:

**Step 1** Right-click the selected table and select **Set Tablespace**.

**Set Tablespace** dialog box is displayed that prompts you to select the new tablespace.

**Step 2** Select the tablespace from the drop-down list and click **OK**. The selected table will be moved to the new tablespace. To create a new tablespace, refer to **Working with Tablespaces**.

The status bar displays the status of the completed operation.

**NOTE**

- This operation will not be successful if you do not have the required access.
- This operation is not available for Partition table.

---End

4.16.6.9 Setting the Schema

Follow the steps to set the schema:

**Step 1** Right-click the selected table and select **Set Schema**.

**Set Schema** dialog box is displayed that prompts you to select the new schema for the selected table.

**Step 2** Select the schema name from the drop-down list and click **OK**. The selected table will be moved to the new schema.

The status bar displays the status of the completed operation.

**NOTE**

- This operation is not supported for Partition ORC tables.
- If the required schema contains a table with the same name as the current table, then Data Studio does not allow setting the schema for the table.

---End
4.16.6.10 Dropping a Table

Individual or batch drop can be performed on tables. Refer to Dropping Batch of Objects section for batch drop.

This operation removes the complete table structure (including the table definition and index information) from the database and you have to re-create this table once again to store data.

Follow the steps to drop the table:

**Step 1** Right-click the selected table and select **Drop Table**.

Data Studio prompts you to confirm this operation.

**Step 2** In the confirmation dialog box, click **OK** to complete the operation successfully.

The status bar displays the status of the completed operation.

End

4.16.6.11 Viewing Table Properties

Follow the steps to view the properties of the table:

**Step 1** Right-click the selected table and select **Properties**.

Data Studio displays the properties (General, Columns, Constraints, and Index) of the selected table in different tabs.

The following table lists the operations that can be performed on each tab along with edit and refresh of data operation. Edit operation is performed by double-clicking on the cell.

<table>
<thead>
<tr>
<th>Tab Name</th>
<th>Operations Allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>Save, Cancel, and Copy</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong> Only Table Description field can be modified.</td>
</tr>
<tr>
<td>Columns</td>
<td>Add, Delete, Save, Cancel, and Copy</td>
</tr>
<tr>
<td>Constraints</td>
<td>Add, Delete, Save, Cancel, and Copy</td>
</tr>
<tr>
<td>Index</td>
<td>Add, Delete, Save, Cancel, and Copy</td>
</tr>
</tbody>
</table>

Refer to Editing Table Data section for more information on edit, save, cancel, copy, paste, refresh operations.

---

**NOTICE**

When viewing table data, Data Studio automatically adjusts the column widths for table view. Users can resize the columns as needed. If the text contents of a cell exceeds the total available display area, then resizing the cell column may cause DS to become unresponsive.
Individual property window is displayed for each table.
If the property of a table is modified for the table that is already opened, then refresh and open the properties of the table again to view the updated information on the same opened window.
If the content of the column have spaces between the words, then word wrap is applied to fit the column within the display area. Word wrap is not applied if the content does not have any spaces between the words.
The size of the column is determined by the maximum content length column.
Any change made to the table properties from Object Browser will be reflected after refreshing the Properties tab.
Paste operation is not allowed in Data Type column.

4.16.6.12 Grant/Revoke Privilege

Follow the steps to grant/revoke privilege:

Step 1 Right-click selected regular/partition table and select Grant/Revoke.

The Grant/Revoke dialog is displayed.

Step 2 Refer to Grant/Revoke Privilege - Regular/Partition Table section to grant/revoke privilege.

4.16.7 Managing Table Data

4.16.7.1 Overview

This section describes how to manage table data.

4.16.7.2 Exporting Table DDL

Follow the steps to export the table DDL:

Step 1 In the Object Browser pane, right-click the selected table and select Export DDL.

The Data Studio Security Disclaimer dialog box is displayed.

Step 2 Click OK.

The Save As dialog box is displayed.

Step 3 In the Save As dialog box, select the location to save the DDL and click Save. The status bar displays the progress of the operation.
To cancel the export operation, double-click the status to open the Progress View tab and click ✗.

- The exported file name will not be the same as table name, if the table name contains characters which are not supported by Windows.
- Microsoft Visual C runtime file (msvcrt100.dll) is required to complete this operation. Refer to Troubleshooting section for more information.

The Export message and status bar displays the status of the completed operation.

<table>
<thead>
<tr>
<th>Database Encoding</th>
<th>File Encoding</th>
<th>Supports Exporting DDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTF-8</td>
<td>UTF-8</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>GBK</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>LATIN1</td>
<td>Yes</td>
</tr>
<tr>
<td>GBK</td>
<td>GBK</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>UTF-8</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>LATIN1</td>
<td>No</td>
</tr>
<tr>
<td>LATIN1</td>
<td>LATIN1</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>GBK</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>UTF-8</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Multiple objects can be selected to export DDL on regular and partition tables. Refer to Batch Export section for list of objects not supported for export DDL operation.

----End

4.16.7.3 Exporting Table DDL and Data

Exporting the table DDL and data exports the following:

- DDL of the table.
- Columns and rows of the table.

Follow the steps below to export the table DDL:

**Step 1** In the Object Browser pane, right-click the selected table and select Export DDL and Data.

The Data Studio Security Disclaimer dialog box is displayed.

**Step 2** Click OK.

The Save As dialog box is displayed.
Step 3 In the Save As dialog box, select the location to save the DDL and click Save. The status bar displays the progress of the operation.

NOTE

- To cancel the export operation, double-click the status to open the Progress View tab and click ☒.
- The exported file name will not be the same as table name, if the table name contains characters which are not supported by Windows.
- Microsoft Visual C runtime file (msvcrt100.dll) is required to complete this operation. Refer to Troubleshooting section for more information.

The Export message and status bar displays the status of the completed operation.

<table>
<thead>
<tr>
<th>Database Encoding</th>
<th>File Encoding</th>
<th>Supports Exporting DDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTF-8</td>
<td>UTF-8</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>GBK</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>LATIN1</td>
<td>Yes</td>
</tr>
<tr>
<td>GBK</td>
<td>GBK</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>UTF-8</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>LATIN1</td>
<td>No</td>
</tr>
<tr>
<td>LATIN1</td>
<td>LATIN1</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>GBK</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>UTF-8</td>
<td>Yes</td>
</tr>
</tbody>
</table>

NOTE

Multiple objects can be selected to export DDL and data on regular and partition tables. It exports columns, rows, indexes, constraints, and partitions. Refer to Batch Export section for list of objects not supported for export DDL and Data operation.

4.16.7.4 Exporting Table Data

Follow the steps below to export table data:

Step 1 Right-click the selected table and select Export Table Data.

The Export Table Data dialog box is displayed with the following options:

- **Format** - Table data can be exported either as excel (xlsx/xls), CSV, Text or binary format. By default Excel (xlsx) is selected.
- **Include Header** - This option is available for CSV and Text files. If selected, it will include the column headers. By default, this option is selected when
exporting to CSV or Text file, although it is not a mandatory field. This field will be disabled for excel (xlsx/xls) and binary format.

- **Quotes** - Use this option to define the quote character. You should enter only single byte character for this field. Quote character should not be same as delimiter. For CSV and Text format, by default this field is enabled, although it is not a mandatory field. This field will be disabled for excel (xlsx/xls) and binary format.
  - If table data value has delimiter in their values, then it will use the character mentioned in this field.
  - If the Quote character is present in value, then that character will be escaped with same quoted character.
  - If result value has multiline values, then it will be quoted with quoted character.

- **Escape** - Use this option to define the escape value. You should enter only single byte character for this field. Escape value should not be same as quote character. For CSV and Text format, by default this field is enabled, although it is not a mandatory field. This field will be disabled for excel (xlsx/xls) and binary format.

- **Replace NULL with** - Use this option to replace null value in the table with string. New line or carriage return characters are non-acceptable values for this field. Maximum of 100 characters can only be entered in this field. This field value must be different from delimiter and quote values. For CSV and Text format, by default this field is enabled, although it is not a mandatory field. This field will be disabled for excel (xlsx/xls) and binary format. It is mandatory to enter a value when Other field is selected.

- **Encoding** - The Encoding field will be pre-populated with the encoding selection made in Preferences > Session Setting tab. This is not a mandatory field.

- **Delimiter** - Use this option to define delimiter. You can select the available delimiter or mention customized delimiter in the Other field in the Delimiter section. For CSV and Text format ",," will be the default delimiter. Maximum of 10 bytes can only be entered in the Other field. For CSV and Text format, by default this field is enabled, although it is not a mandatory field. This field will be disabled for excel (xlsx/xls) and binary format. It is mandatory to enter a value when Other field is selected.

- **All Columns** - Use this option to quick select all columns. By default this is checked. To manually select columns, uncheck this and select columns from the Available Columns list.
  - **Available Columns** - You can use this column to select specific columns to export.
  - **Selected Columns** - This field displays the selected columns that will be exported. The columns can be re-ordered. By default all columns display in this field.

**NOTE**

Refer to Column/Row Size in FAQ section for row and column size supported by xlsx and xls.

- **File Name** - Use this option to specify the name to save the exported file. By default, the table name is displayed in this field.
File name follows Windows file naming convention.

- **Output Path** - Use this option to select the location to save the exported file. The **Output Path** field is auto-populated with the selected path.

- **Security Disclaimer** - The security disclaimer is mentioned in this section, and you should read and agree, to continue with the export operation.
  - **I Agree** - By default this field is selected. You cannot proceed further if this field is not checked.
  - **Do not show again** - You can select this field to hide the Security Disclaimer for subsequent export table data operation for current logged instance of Data Studio.

**NOTE**

- String, double, date, calendar, and boolean data type will be stored as is in Excel. All other data type will be converted into string and stored in Excel.
- For Excel, export if the cell size is beyond 32767, then exported cell data will be truncated.

**Step 2** Complete the required fields and click **OK**.

The **Save As** dialog box is displayed.

**Step 3** Click **Save** to save the exported data in the selected format. The status bar displays the progress of the operation.

The **Data Exported Successfully** dialog box and status bar displays the status of the completed operation.

**NOTE**

- If the disk is full while exporting the table, then Data Studio displays an I/O error. Perform the following operations to resolve this error:
  1. Click **OK** to close the connection profile.
  2. Clean the disk.
  3. Re-establish the connection and export the table data.

- The exported file name will not be the same as table name, if the table name contains characters which are not supported by Windows.

---End

**Canceling the export table data operation**

Follow the steps to cancel the export table data operation:

**Step 1** Double-click the status bar to open the **Progress View** tab.

**Step 2** In the **Progress View** tab, click 🗑️.

**Step 3** In the **Cancel Operation** dialog box, click **Yes**.

The **Messages** tab and status bar displays the status of the cancelled operation.

---End
4.16.7.5 Showing DDL

Follow the steps to show DDL query of the table:

**Step 1** Right-click the selected table and select **Show DDL**.

The DDL of the selected table is displayed.

**NOTE**

- A new terminal is opened each time the **Show DDL** operation is executed.
- Microsoft Visual C runtime file (msvcr100.dll) is required to complete this operation. Refer to **Troubleshooting** section for more information.

<table>
<thead>
<tr>
<th>Database Encoding</th>
<th>File Encoding</th>
<th>Supports Show DDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTF-8</td>
<td>UTF-8</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>GBK</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>LATIN1</td>
<td>Yes</td>
</tr>
<tr>
<td>GBK</td>
<td>GBK</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>UTF-8</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>LATIN1</td>
<td>No</td>
</tr>
<tr>
<td>LATIN1</td>
<td>LATIN1</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>GBK</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>UTF-8</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### End

4.16.7.6 Importing Table Data

Prerequisites to import table data are:

- If the source import file does not match with the destination import table definition, then you must modify the properties of the destination table in the **Import Table Data** dialog box. Additional columns will be inserted with default value.
- You should know the export properties of the file that you are importing like delimiter, quote, and escape character and so on. Export properties saved during export operation cannot be changed while importing the file.

Follow the steps to import table data:

**Step 1** Right-click the selected table and select **Import Table Data**.

Data Studio displays the **Import Table Data** dialog box with the following options:

- **Import Data File** - This field displays the file path of the imported file. Use the **Browse** button to select different file.
- **Format** - Table data can be imported as CSV, Text or binary format. By default CSV is selected.
Include Header - Use this option if the import file has column header. For CSV and Text format, by default this field is selected, although it is not a mandatory field. This field will be disabled for binary format.

Quotes - You should enter only single byte character for this field. Quote character should not be same as delimiter and null parameter. For CSV and Text format, by default this field is selected, although it is not a mandatory field. This field will be disabled for binary format.

Escape - You should enter only single byte character for this field. If escape value is same as quote value, then escape value will be replaced with '\0'. For CSV and Text format, by default this field is selected with value as double quotation mark, although it is not a mandatory field. This field will be disabled for binary format.

Replace with Null - You can use this field to replace null value in the table with string. The same null string used while exporting should be used while importing data and this need to be explicitly mentioned. For CSV and Text format, by default this field is selected, although it is not a mandatory field. This field will be disabled for binary format.

Encoding - The Encoding field will be pre-populated with the encoding selection made in Preferences > Session Setting tab. This is not a mandatory field.

Delimiter - You can select the available delimiter or mention customized delimiter in the Other field in the Delimiter section. For CSV and Text format, "," will be the default delimiter. This field value should not be same as Quote and Replace Null with field values. For CSV and Text format, by default this field is selected, although it is not a mandatory field. This field will be disabled for binary format. It is mandatory to enter a value when Other field is selected.

All Columns - Use this option to quick select all columns. By default this field is selected. To manually select columns, un-check this and unselected columns from the Selected Columns list.

Step 2 Click the Browse button from the Import Data File field. The Open dialog box is displayed.

Step 3 In the Open dialog box, select the file to import and click Open.

Step 4 Complete the required fields and click OK.

The status bar displays the progress of the operation. The imported data is appended to the existing table data.

The Data Imported Successfully dialog box and status bar displays the status of the completed operation.

----End

Canceling the import table data operation

Follow the steps to cancel the import table data operation:
Step 1 Double-click the status bar to open the Progress View tab.

Step 2 In the Progress View tab, click .

Step 3 In the Cancel Operation dialog box, click Yes.

The Messages tab and status bar displays the status of the canceled operation.

--- End

4.16.7.7 Viewing Table Data

Follow the steps to view table data:

Step 1 Right-click the selected table and select View Table Data.

The View Table Data tab is displayed where you can view the table data information.

Toolbar menu in the View Table Data window:

<table>
<thead>
<tr>
<th>Toolbar Name</th>
<th>Toolbar Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy</td>
<td>![Copy Icon]</td>
<td>Click the icon to copy selected content from View Table Data window to clipboard. Shortcut key - Ctrl+C.</td>
</tr>
<tr>
<td>Advanced Copy</td>
<td>![Advanced Copy Icon]</td>
<td>Click the icon to copy content from result window to clipboard. Results can be copied to include the row number and/or column header. Refer to View Query Results to set this preference. Shortcut key - Ctrl+Shift+C.</td>
</tr>
<tr>
<td>Show/Hide Search bar</td>
<td>![Show/Hide Search bar Icon]</td>
<td>Click the icon to display/hide the search text field. This is a toggle button.</td>
</tr>
<tr>
<td>Encoding</td>
<td>-</td>
<td>Refer to Execute SQL Queries section for information on encoding selection.</td>
</tr>
</tbody>
</table>

Icons in Search field:

<table>
<thead>
<tr>
<th>Icon Name</th>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search</td>
<td>![Search Icon]</td>
<td>Click the icon to search the table data displayed based on the criteria defined. Search text are case insensitive.</td>
</tr>
<tr>
<td>Clear Search Text</td>
<td>![Clear Search Text Icon]</td>
<td>Click the icon to clear the search text entered in the search field.</td>
</tr>
</tbody>
</table>

Refer to Execute SQL Queries section for column reordering and sort option.
- **Query Submit Time** - Provides the query submitted time.
  - Number of rows fetched with execution time is displayed. The default number of rows is displayed. If there are additional rows to be fetched, then it will be denoted with the word "more". You can scroll to the bottom of the table to fetch and display all rows.

---

**NOTICE**

- When viewing table data, Data Studio automatically adjusts the column widths for an optimal table view. Users can resize the columns as needed. If the text contents of a cell exceeds the total available display area, then resizing the cell column may cause DS to become unresponsive.
- When the data in a table cell is more than 1000 characters, it will appear trimmed up to 1000 characters with "..." at the end.
  - If the user copies the data from a cell in a table or Result tab and pastes it on any editor (such as SQL terminal/PLSQL source editor, notepad or any other external editor application), the entire data is pasted.
  - If the user copies the data from a cell in a table or Result tab and pastes it on an editable cell (same or different), the cell shows only the first 1000 characters with "...
  - When the table/Result tab data is exported, the exported file contains the whole data.

---

**NOTE**

- Individual table data window is displayed for each table.
- If the data of the table that is already opened is modified, then refresh and open the table data again to view the updated information on the same opened window.
- While the data is loading a message displays at the bottom stating “fetching”.
- If the content of the column have spaces between the words, then word wrap is applied to fit the column within the display area. Word wrap is not applied if the content does not have any spaces between the words.
- Select part of cell content and press **Ctrl+C** or click ![copy](image) to copy selected text from a cell.
- The size of the column is determined by the maximum content length column.
- You can save preference to define:
  - Number of records to be fetched.
  - Column width
  - Copy option from result set.
  - Refer to **Query Results** for more information.

---

4.16.7.8 Editing Table Data

Follow the steps to edit table data:

**Step 1** Right-click the selected table and select **Edit Table Data**.
The **Edit Table data** tab is displayed.

Refer to **Viewing Table Data** section for description on copy and search toolbar options.

----End

Data Studio validates only the following data types entered into cells:

- Bigint
- bit
- boolean
- char
- date
- decimal
- double
- float
- integer
- numeric
- real
- smallint
- time
- time with time zone
- time stamp
- time stamp with time zone
- tinyint
- varchar

Editing of array type data type is not supported.

Any related errors during this operation reported by database are displayed in Data Studio. Time with time zone and timestamp with time zone columns are non-editable columns.

You can perform the following operations in the **Edit Table Data** tab:

- **Insert**
- **Delete**
- **Update**
- **Copy**
- **Paste**

**Insert**

Follow the steps to insert a row:

**Step 1** Click  to insert a row.

**Step 2** Double-click the cell to modify and enter the required details in the row.
Step 3  Click to save changes.

The Edit Table Data tab status bar shows the Query Submit Time, Number of rows fetched, Execution time and Status of the operation.

**NOTICE**

Data Studio updates rows identified by the unique key. If a unique key is not identified for a table and there are identical rows, then an update operation made on one of the rows will affect all identical rows. Refresh the Edit Table Data tab to view the updated rows.

**NOTE**

- Changes to cells in a row that are not saved are highlighted in green. Once saved the color resets to default color.
- Unsaved records are highlighted in red. The number of successful and failed records are displayed in the status bar of the Edit Table Data tab.
- Clicking Save either saves all the valid changes or does not save anything if there are invalid changes. Refer to Edit Table Data to set the behavior of save operation.

Step 4  Click to roll back the changes that are not saved. Set the preference to define:

- Number of records to be fetched
- Column width
- Copy option from result set

Refer to Query Results for more information.

----End

Data Studio allows you to edit the distribution key column only for a new row.

Delete

Follow the steps to delete a row:

Step 1  Click the row header of the row to be deleted.

Step 2  Click to delete a row.

Step 3  Click to save changes.

Define unique key dialog box is displayed.

Step 4  Click the required option:

- **Use All Columns**
  
  Click Use All Columns to define all columns as unique key.

- **Custom Unique Key**

  a. Click Custom Unique Key to define selected columns as unique key.
  
  b. Define Unique Key dialogue box is displayed.
c. Select the required columns and click OK.

- **Cancel**
  Click Cancel to modify the information in Edit Table Data tab.
  The Edit Table Data tab status bar shows the Query Submit Time, Number of rows fetched, Execution time and Status of the operation.
  Select Remember the selection for this window option to hide the unique definition window from displaying while continuing with the edit table data operation. Click \(\square\) from Edit Table Data toolbar to clear previously selected unique key definition and display unique definition window again.

**NOTE**

- Deleted rows that are not saved are highlighted in red. Once saved the color resets to default color.
- Unsaved records are highlighted in red. The number of successful and failed records are displayed in the status bar of the Edit Table Data tab.
- Clicking Save either saves all the valid changes or does not save anything if there are invalid changes. Refer to Edit Table Data to set the behavior of save operation.

**Step 5** Click \(\text{.rollback}\) to roll back the changes that are not saved.

**Step 6** Refresh the table data to view deleted duplicate rows.

----End

**Update**

Follow the steps to update cell data:

**Step 1** Double-click the cell to update the contents of the cell.

**Step 2** Click \(\text{save}\) to save changes.

Define unique key dialog box is displayed.

**Step 3** Click the required option:

- **Use All Columns**
  Click Use All Columns to define all columns as unique key.

- **Custom Unique Key**
  a. Click Custom Unique Key to define selected columns as unique key.
  b. Define Unique Key dialogue box is displayed.
  c. Select the required columns and click OK.

- **Cancel**
  Click Cancel to modify the information in Edit Table Data tab.
  The status bar shows the Execution Time and Status of the operation.
  Select Remember the selection for this window option to hide the unique definition window from displaying while continuing with the edit table data operation. Click \(\square\) from Edit Table Data toolbar to clear previously selected unique key definition and display unique definition window again.
NOTE

- Changes to cells in a row that are not saved is highlighted in green. Once the record is saved the color resets to default color.
- Unsaved records are highlighted in red. The number of successful and failed records are displayed in the status bar of the Edit Table Data tab.
- Clicking Save either saves all the valid changes or does not save anything if there are invalid changes. Refer to Edit Table Data to set the behavior of save operation.

Step 4  Click to roll back the changes that are not saved.

Step 5  Refresh the table data to view deleted duplicate rows.

--- End

During edit operation, Data Studio does not allow you to edit the distribution key column as it is used by the DB to locate data in the database cluster.

Copy

You can copy data from the Edit Table Data tab.

Follow the steps below to copy data:

Step 1  Select the cell(s) and click (Copy) or (Advanced Copy).

Refer to Execute SQL Queries section to understand the difference between copy and advanced copy.

NOTE

- Data can be copied to include the row number and/or column header. Refer to Query Results to set this preference.
- Select part of cell content and press Ctrl+C or click to copy selected text from a cell.

--- End

Paste

You can copy data from a CSV file and paste it into cells in the Edit Table Data tab to insert and update records. If you paste onto existing cell data, the data is overwritten with the new data from the CSV file.

Follow the steps to paste data into a cell:

Step 1  Copy data from CSV file.

Step 2  Select the cell(s) and click .

Step 3  Click to save changes.

Define unique key dialogue box is displayed.

Step 4  Click the required option:

- Use All Columns
Click **Use All Columns** to define all columns as unique key.

- **Custom Unique Key**
  a. Click **Custom Unique Key** to define selected columns as unique key.
  b. **Define Unique Key** dialogue box is displayed.
  c. Select the required columns and click **OK**.

- **Cancel**
  Click **Cancel** to modify the information in **Edit Table Data** tab.

  The status bar shows the **Execution Time** and **Status** of the operation.

  Select **Remember the selection for this window** option to hide the unique definition window from displaying while continuing with the edit table data operation. Click [ ] from **Edit Table Data** toolbar to clear previously selected unique key definition and display unique definition window again.

  **NOTE**

  - The number of copied cells from CSV must match the number of cells selected in the **Edit Table Data** tab to paste the data.
  - Use the [ ] to roll back the changes that are not saved.
  - Changes to cells in a row that are not saved is highlighted in green. Once saved the color resets to default color.
  - Failed unsaved records are highlighted in red. The number of successful and failed records are displayed in the status bar of the **Edit Table Data** tab.
  - Clicking **Save** either saves all the valid changes or does not save anything if there are invalid changes. Refer to **Edit Table Data** to set the behavior of save operation.

--- End

During paste operation, Data Studio does not allow you to edit the distribution key column as it is used by the DB to locate data in the database cluster.

**NOTE**

Empty cells are shown as [NULL]. Empty cell in **Edit Table Data** tab can be searched using the Null Values search drop-down.

Refer to Execute SQL Queries section for information on show/hide search bar, sort, column reorder, and encoding options.

### 4.16.8 Editing Temporary Tables

Data Studio allows you to edit temporary tables. Temporary tables are deleted automatically when you close the connection that was used to create the table.

**NOTICE**

Ensure that connection reuse is enabled when you use the SQL Terminal to edit temporary tables. Refer to Managing SQL Terminal Connections for information about enabling SQL Terminal Connection reuse.

Follow the steps to edit a temporary table:
Step 1 Execute a query on the temporary table.

The Result tab displays the results of the SQL query along with the query statement executed.

Step 2 Edit the temporary table from the Result tab. Refer to the Execute SQL Queries section for information on editing the resultset.

4.17 Sequences

4.17.1 Creating Sequence

Follow the steps to create a sequence:

Step 1 In the Object Browser pane, right-click Sequences under the particular schema where you want to create the sequence and select Create Sequence.

The Create New Sequence dialog box is displayed.

Step 2 Provide information to create a sequence:

1. Enter a name in Sequence Name field.

   ☐ NOTE

   Select Case check box to retain the capitalization of the text entered in Sequence Name field. For example, if the sequence name entered is "Employee", then the sequence name is created as "Employee".

2. Enter minimum value in Minimum Value field.

3. Enter the increase step value in Increment By field.

4. Enter maximum value in Maximum Value field.

   ☐ NOTE

   Minimum and Maximum value should be between -9223372036854775808 and 9223372036854775807.

5. Enter start value of the sequence in Start Value field.

6. Enter cache information in Cache field. The cache value denotes the number of sequences stored in the memory for quick access.

7. Select Cycle field to recycle sequences after the number of sequences reaches either the maximum or minimum value.

   ☐ NOTE

   The schema name auto-populates in the Schema field.

8. Select table from Table drop-down.

9. Select column from Column drop-down.

Step 3 Click Finish.

The status bar displays the status of the completed operation.
4.17.2 Grant/Revoke Privilege

Follow the steps to grant/revoke privilege:

**Step 1** Right-click sequences group and select *Grant/Revoke*. The *Grant/Revoke* dialog is displayed.

**Step 2** Select the objects to grant/revoke privilege from *Object Selection* tab and click *Next*.

**Step 3** Select the role from *Role* drop-down in *Privilege Selection* tab.

**Step 4** Select *Grant/Revoke* in *Privilege Selection* tab.

**Step 5** Select/unselect the required privileges in *Privilege Selection* tab. In *SQL Preview* tab, you can view the SQL query automatically generated for the inputs provided.

**Step 6** Click *Finish*.

4.17.3 Working with Sequences

You can perform the following operations on an existing sequence:

- *Grant/Revoke Privilege*
- *Dropping a Sequence*
- *Dropping a Sequence Cascade*

**Dropping a Sequence**

Individual or batch drop can be performed on sequences. Refer to *Dropping Batch of Objects* section for batch drop.

Follow the steps to drop a sequence:

**Step 1** Right-click the selected sequence and select *Drop Sequence*. The *Drop Sequence* dialog box is displayed.

**Step 2** Click *Yes* to drop the sequence. The status bar displays the status of the completed operation.

---End

**Dropping a Sequence Cascade**

Follow the steps to drop the sequence cascade:
Step 1  Right-click the selected sequence and select **Drop Sequence Cascade**.
The **Drop Sequence Cascade** dialog box is displayed.

Step 2  Click **Yes** to drop the sequence cascade.
The status bar displays the status of the completed operation.

--- End

**NOTE**
This is only available for OLAP, not for OLTP.

**Grant/Revoke Privilege**

Follow the steps to grant/revoke privilege:

Step 1  Right-click selected sequence and select **Grant/Revoke**.
The **Grant/Revoke** dialog is displayed.

Step 2  Refer to **Grant/Revoke Privilege** section to grant/revoke privilege.

--- End

### 4.18 Views

#### 4.18.1 Creating a View

Follow the steps to create a new view:

Step 1  Right-click **Views** and select **Create View**.
The DDL template for the view is displayed in the SQL Terminal tab.

Step 2  Edit the DDL as required.

Step 3  Click ![Execute DDL](image) to execute the DDL.

Step 4  Press **F5** to refresh the **Object Browser**.
You can view the new view in the **Object Browser**.

**NOTE**
The status bar does not display message on completion of this operation.

--- End

#### 4.18.2 Grant/Revoke Privilege

Follow the steps to grant/revoke privilege:

Step 1  Right-click views group and select **Grant/Revoke**.
The **Grant/Revoke** dialog is displayed.
Step 2 Select the objects to grant/revoke privilege from Object Selection tab and click Next.

Step 3 Select the role from Role drop-down in Privilege Selection tab.

Step 4 Select Grant/Revoke in Privilege Selection tab.

Step 5 Select/unselect the required privileges in Privilege Selection tab.

In SQL Preview tab, you can view the SQL query automatically generated for the inputs provided.

Step 6 Click Finish.

4.18.3 Working with Views

Views can be created to restrict access to specific rows or columns of a table. A view can be created from one or more tables and is determined by the query used to create the view.

You can perform the following operations on an existing view:

- Exporting the View DDL
- Dropping a View
- Dropping a View Cascade
- Renaming a View
- Setting the Schema for a View
- Viewing the Show DDL
- Setting the Default Value for the View Column
- Viewing the Properties of a View
- Grant/Revoke Privilege

Exporting the View DDL

Follow the steps to export view DDL:

Step 1 Right-click the selected view and select Export DDL.

Data Studio Security Disclaimer dialog box is displayed.

Step 2 Click OK.

The Save As dialog box is displayed.

Step 3 In the Save As dialog box, select the location to save the DDL and click Save. The status bar displays the progress of the operation.
To cancel the export operation, double-click the status to open the Progress View tab and click 

The exported file name will not be the same as view name, if the view name contains characters which are not supported by Windows.

Multiple objects can be selected to export the view DDL. Refer to Batch Export section for list of objects not supported for export view DDL operation.

The Export message and status bar displays the status of the completed operation.

---End

### Dropping a View

Individual or batch drop can be performed on views. Refer to Dropping Batch of Objects section for batch drop.

Follow the steps to drop the view:

**Step 1** Right-click the selected view and select Drop View.

The Drop View dialog box is displayed.

**Step 2** Click Yes to drop the view.

The status bar displays the status of the completed operation.

---End

### Dropping a View Cascade

Follow the steps to drop the view and its dependent database objects:

**Step 1** Right-click the selected view and select Drop View Cascade.

The Drop View dialog box is displayed.
Step 2  Click Yes to drop the view and its dependent database objects. The status bar displays the status of the completed operation.

----End

Renaming a View

Follow the steps to rename the view:

Step 1  Right-click the selected view and select Rename View.
The Rename View dialog box is displayed.

Step 2  Enter the required name for the view and click OK. You can view the renamed view in the Object Browser.
The status bar displays the status of the completed operation.

----End

Setting the Schema for a View

Follow the steps to set the schema for the view:

Step 1  Right-click the selected view and select Set Schema.
The Set Schema dialog box is displayed.

Step 2  Select the required schema from the drop-down list and click OK.
The status bar displays the status of the completed operation.
If the required schema contains a view with the same name as the current view, then Data Studio does not allow setting the schema for the view.

----End

Viewing the Show DDL

Follow the steps to view the DDL of the view:

Step 1  Right-click the selected view and select Show DDL.
The DDL is displayed in a new SQL Terminal tab. You must refresh the Object Browser to view the latest DDL.

----End

Setting the Default Value for the View Column

Follow the steps to set the default value for a column in the view:

Step 1  Right-click the selected column name under the view and select Set View Column Default Value.
A dialog box with the current default value (if it is set) is displayed which prompts you to provide the default value.
Step 2  Enter the value and click OK.
Data Studio displays the status of the operation in the status bar.

----End

Viewing the Properties of a View

Follow the steps to view the properties of the View:

Step 1  Right-click the selected View and select Properties.
The properties (General and Columns) of the selected View is displayed in different tabs.

NOTE
If the property of a View is modified that is already opened, then refresh and open the properties of the View again to view the updated information on the same opened window.

----End

Grant/Revoke Privilege

Follow the steps to grant/revoke privilege:

Step 1  Right-click selected view and select Grant/Revoke.
The Grant/Revoke dialog is displayed.
Step 2  Refer to Grant/Revoke Privilege section to grant/revoke privilege.

----End

4.19 Tablespace

4.19.1 Creating a Tablespace

You can create tablespaces to optimize performance of database objects.
After creating a tablespace, it will be available in the Create New table wizard.
For more information, refer to Creating Regular Table.

Follow the steps to create a new tablespace for DWS database:

Step 1  Right-click Tablespace and select Create Tablespace. The Create Tablespace dialog box is displayed.
Step 2  Enter the following information to create the tablespace:

NOTE
You need to fill all the mandatory parameters that are marked with asterisk (*) to complete the operation successfully.

- Name: Enter the name of the tablespace. For example, New_Tablespace.
- Location: Enter the path to store the tablespace on the database. For example, /home/user1
One path is limited to only one tablespace. Access permissions to the path must be set by the user.

- **Unlimited Size**: Select this check box to set unlimited maximum size of the tablespace.

**NOTE**

Once the **Unlimited Size** field is checked, the **Max Size** field becomes non-editable.

- **Max Size**: Enter the maximum size of the tablespace. The supported range is 1 KB - 9,007,199,254,740,991 KB. The only acceptable value in this field is positive whole number.

Select an option from the **Max Size** drop-down list. The options available are:
- **KB**: Specifies the **Max Size** in kilobytes.
- **MB**: Specifies the **Max Size** in megabytes.
- **GB**: Specifies the **Max Size** in gigabytes.
- **TB**: Specifies the **Max Size** in terabytes.
- **PB**: Specifies the **Max Size** in petabytes.

- **File System**: Select the type of file system.
  - Select **General** to create the tabular type of tablespace.
  - Select **HDFS** to create the Hadoop Distributed File System (HDFS) type of tablespace.

- **Address**: Enter the Name, Node IP address (IPv4) of the HDFS cluster and port number of the active and standby nodes.
  For example, xx.xx.xx.xx:xxxx

- **HDFS Configuration File Path**: Enter the path of the HDFS cluster configuration file. This is a mandatory field for HDFS file system.

- **HDFS Data Store Path**: Enter the HDFS data storage path. This is a mandatory field for HDFS file system.

- **Sequential Page Cost**: Sets the optimizer's estimated cost of a disk page fetch that is part of a series of sequential fetches. Enter the sequential read page overhead. The supported range is 0 - 1.79769e+308 (double byte). The default value is 1 which is also the recommended value. The acceptable values are either positive whole number or positive decimals with one decimal point.

- **Random Page Cost**: Sets the optimizer's estimated cost of a non-sequentially-fetched disk page. Enter the random read page overhead. The supported range is 0 - 1.79769e+308 (double byte). The default value is 4 which is also the recommended value. The acceptable values are either positive whole number or positive decimals with one decimal point.
Although the server allows to set the value of Random Page Cost to less than that of Sequential Page Cost, it is not physically sensitive to do so. However, setting them equal makes sense if the database is entirely cached in RAM, because in that case there is no penalty for fetching pages out of sequence. Also, in a heavily-cached database you must lower both values relative to the CPU parameters, since the cost of fetching a page already in RAM is much smaller than it would normally be.

---

If the server does not support HDFS file system, then Address, HDFS Configuration File Path, and HDFS Data Store Path, fields becomes non-editable.

**Step 3** Click OK. You can view the new tablespace in the **Object Browser**. The status bar displays the status of the completed operation.

---

### 4.19.2 Working with Tablespaces

You can perform the following operations on an existing tablespace:

- Setting Tablespace Options
- Setting Maximum Tablespace Size
- Dropping a Tablespace
- Viewing the Tablespace DDL

**Renaming a Tablespace**

Follow the steps to rename a tablespace for Analytical cluster, Ultra-highly concurrent cluster, and DWS database:

**Step 1** Right-click the selected tablespace and select **Rename Tablespace**. The **Rename Tablespace** dialog box is displayed.

**Step 2** Enter the new name for the tablespace and click **OK**. The status bar displays the status of the completed operation.

---

**NOTE**

System tablespaces are displayed in the **Object Browser** by default.
You can view the renamed tablespace in the **Object Browser**.

### Setting Tablespace Options

Follow the steps to set tablespace options:

**Step 1** Right-click the selected tablespace and select **Set Tablespace Option**.

The **Set Tablespace Option** dialog box is displayed.

**Step 2** Enter the **Random Page Cost** and **Sequential Page Cost** and click **OK**.

The status bar displays the status of the completed operation.

### Setting Maximum Tablespace Size

Follow the steps to set the maximum size of the tablespace:

**Step 1** Right-click the selected tablespace and select **Set Tablespace MaxSize**.

The **Set Tablespace MaxSize** dialog box is displayed.

**Step 2** To set an unlimited value for the maximum size of the tablespace, select the **Unlimited Size** check box.

To set a custom value for the maximum size of the tablespace, enter the maximum size of the tablespace in the **Max Size** text box. The supported range is 1 KB - 9,007,199,254,740,991 KB. Select an option from the **Max Size** drop-down list. The options available are:

- **KB**: Specifies the **Max Size** in kilobytes.
- **MB**: Specifies the **Max Size** in megabytes.
- **GB**: Specifies the **Max Size** in gigabytes.
- **TB**: Specifies the **Max Size** in terabytes.
- **PB**: Specifies the **Max Size** in petabytes.

**Step 3** Click **OK**.

The status bar displays the status of the completed operation.

### Dropping a Tablespace

Follow the steps to drop the tablespace for Analytical cluster, Ultra-highly concurrent cluster, and DWS database:

**NOTE**

- Data Studio does not allow dropping of system tablespace(s).
- Data studio does not allow batch drop of tablespaces.

**Step 1** Right-click the selected tablespace and select **Drop Tablespace**.
The **Drop Tablespace** dialog box is displayed.

**Step 2**  Click OK to drop the tablespace.

The status bar displays the status of the completed operation.

----End

### Viewing the Tablespace DDL

Follow the steps to view the DDL of the tablespace:

**Step 1**  Right-click the selected tablespace and select **Show DDL**.

The tablespace DDL is displayed in a new **SQL Terminal** tab with the terminal name format as 'tablespacename@servername'. You must refresh the **Object Browser** to view the latest DDL.

----End

### 4.20 Users/Roles

#### 4.20.1 Create User/Role

A database is used by many users, and the users are grouped for management convenience. A database role can be one or a group of database users.

Users and roles have similar concepts in databases. In practice, you are advised to use a role to manage permissions rather than to access databases.

**Users** - They are set of database users. These users are different from operating system users. These users can assign privileges to other users to access database objects.

**Role** - This can be considered as a user or group based on the usage. Roles are at cluster level, and hence applicable to all databases in the cluster.

#### 4.20.2 Working with Users/Roles

You can perform the following operations on an existing user/role:

- **Dropping a User/Role**
- **Viewing/Editing User/Role Properties**
- **Viewing the User/Role DDL**

**Dropping a User/Role**

Follow the steps to drop a user/role:

**Step 1**  Right-click the selected user/role and select **Drop User/Role**.

The **Drop User/Role** dialog box is displayed.

**Step 2**  Click Yes to drop the user/role.
The status bar displays the status of the completed operation.

---End

Viewing/Editing User/Role Properties

Follow the steps to view the properties of user/role:

**Step 1** Right-click the selected user/role and select **Properties**.

Data Studio displays the properties (General, Privilege, and Membership) of the selected user/role in different tabs.

Editing of properties can be performed. OID is non-editable field.

Refer to **Editing Table Data** section for information on edit, save, cancel, copy, and refresh operations.

---End

Viewing the User/Role DDL

Follow the steps to view the DDL of the user/role:

**Step 1** Right-click the selected user/role and select **Show DDL**.

The user/role DDL is displayed in a new **SQL Terminal** tab. You must refresh the **Object Browser** to view the latest DDL.

---End

4.21 SQL Terminal

4.21.1 Opening SQL Terminals

You can open multiple **SQL Terminal** tabs in Data Studio. You can use this feature to work with SQL queries when the current **SQL Terminal** is executing a query.

Follow the steps below to open a new SQL Terminal:

You can also open multiple SQL terminals on different connection profiles.

**Step 1** In the **Object Browser** pane, right-click the selected database and select **Open Terminal** or click on the toolbar or Ctrl+T shortcut key to open new SQL terminal.

The new **SQL Terminal** tab is displayed.

---End
NOTE

- Data Studio supports a maximum of 100 SQL terminals and tabs in total. Each SQL Terminal will have multiple Result and one Messages tab based on the number of times a query is executed. If the connection with the database is lost, then the corresponding SQL Terminals are not disabled.
- Restoring individual SQL Terminal or tabs is not possible. The restore operation restores the complete set of minimized SQL Terminals and tabs.
- Data Studio resets the numbering counter of SQL Terminal after all terminals are closed.
- Data Studio resets the numbering counter of Resultset after all the tabs are closed.
- Data Studio resets the numbering counter for show DDL Tablespace, show DDL Users/Roles, Batch Drop tab, Result tab and Execution Plan tab.

Errors and warnings are displayed which do not have accompanying results in the status bar. Results of successful executions are displayed in the Result tab.

Follow the steps to open a new SQL Terminal on a different connection profile:

**Step 1** On the toolbar, select the required connection from the connection profile drop-down list.

**Step 2** In the Object Browser pane, right-click the selected database in the connection profile and select Open Terminal or click ☑ on the toolbar. The new SQL Terminal tab is displayed.

The new SQL Terminal tab is named as <database_name>@<connection_profile>(<tab_number>). For example, postgres@IDG_1(2). The tab number is updated for each new SQL Terminal tab of the connection profile.

----End

**Managing Right-Click Option On Result Window**

This feature allows to copy, export cell data to excel files and generate SQL files of queries as well.

Right-click on the result window after the result of the SQL query is shown. Right-click Menu is displayed as following:
Follow the steps for including row number and column header in Result Set:

**Step 1** Click **Settings** on Menu bar of Data Studio.

**Step 2** Select **Preferences**.

**Step 3** Expand **Result Management** and select **Query Results**.

**Step 4** Under **Result Advanced Copy** option check **Include column header** and **Include row number** boxes.

----End

Feature description of the menu is as following:

**Table 4-17** Right-click Menu

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Sub Menu</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy Data</td>
<td>Copy</td>
<td>Copies the selected cell data.</td>
</tr>
<tr>
<td></td>
<td>Advanced Copy</td>
<td>Copies the selected cell data with row number and column header as per the preference setting.</td>
</tr>
<tr>
<td>Copy to Excel</td>
<td>Copy as xls</td>
<td>Exports the selected cell data in xls format. Maximum 64k Rows and 256 Columns can be exported.</td>
</tr>
<tr>
<td></td>
<td>Copy as xlsx</td>
<td>Exports the selected cell data in xlsx format. Maximum 1M Rows can be exported.</td>
</tr>
<tr>
<td>Export</td>
<td>Current Page</td>
<td>Exports the table data of the current page.</td>
</tr>
<tr>
<td></td>
<td>All Pages</td>
<td>Exports the entire table content.</td>
</tr>
<tr>
<td>Generate SQL</td>
<td>Selected Line</td>
<td>Generate the SQL file of the selected data in logical insert statement.</td>
</tr>
<tr>
<td></td>
<td>Current Page</td>
<td>Generates a SQL file of the current page data in logical insert statement.</td>
</tr>
<tr>
<td></td>
<td>All Pages</td>
<td>Generates a SQL file of entire table data in logical insert statement.</td>
</tr>
<tr>
<td>Set Null</td>
<td>-</td>
<td>Sets a cell data to null.</td>
</tr>
<tr>
<td>Search</td>
<td>-</td>
<td>Searches the selected cell data and displays all the data that matches the search condition.</td>
</tr>
</tbody>
</table>
NOTE
Generated SQLs are not valid for result sets derived from queries with Joins, Expressions, Views, Set operators, Aggregate functions, GROUP BY clause and column aliases.

Viewing Text Mode In Result Set Tab
This new feature in Data Studio enables you to view the data in text mode in resultset tab.

Apart from having the grid view, the text mode view provides you with two features: copy and search.

Step 1 Click icon to obtain the result in text mode.

---End

NOTE
Selecting multiple cell data and searching may show some incorrect results in text mode as all the information is copied in plain text to the search window.

Displaying Execution Progress Bar
When a query is being executed from SQL Terminal, A progress bar is shown associated with particular terminal with dynamic elapsed time. The progress bar disappears as the query execution finishes. The time information alongside the bar displays the duration of the query execution on completion.

An option to cancel the query execution is available alongside the progress bar if required.

Refer to the following image:
The Cancel button now has been removed from the toolbar.
Execution Progress Bar is also shown in compiling/debug of function/procedures in PL/SQL editor.
The time format shown in the progress bar will be as: w hrs x min y sec z ms
For batch execution in SQL Terminal, the progress bar is shown with total elapsed time on completion.

**NOTE**

- The Cancel button now has been removed from the toolbar.
- Execution Progress Bar is also shown in compiling/debug of function/procedures in PL/SQL editor.
- The time format shown in the progress bar will be as: w hrs x min y sec z ms
- For batch execution in SQL Terminal, the progress bar is shown with total elapsed time on completion.

**Debug Time Consumption**

Data Studio displays a status bar during debugging, that shows the Last Operation Time consumed for executing statement in each debug operation and also the Total Debug Time.

While Debug operation, based on the each operation performed, the Last Operation Time and Total Debug Time provided in terminal status bar gets updated. Total Debug Time is the cumulative sum of Last Operation Time.

This provides an easy way to find out huge time consuming statement of debug object.

Refer to the following image:
● Debug is applicable for both functions and procedures.
● It is only applicable for OLAP.

4.21.2 Managing SQL Query Execution History

Data Studio allows viewing and managing frequently used SQL queries. The history of executed SQL queries is maintained only for the SQL Terminal.

Follow the steps to view the SQL history:

Step 1 Click in the SQL Terminal tab.

The SQL History dialog box is displayed.

---End

NOTE

SQL history scripts are not encrypted.

The number of queries saved in the SQL History dialog box is based on the value defined in Preferences > Editor > SQL History pane. Refer to the SQL History section to modify the SQL History count. Data Studio overwrites the older queries into the SQL history after the list is full. The executed query is automatically stored in the list.

The SQL History dialog box has the following columns:

● **Pin Status** - Displays the pinned status of the queries. Pinned queries will always show on the top and it will not be deleted from the history even when the list is full.

● **SQL Statement** - Displays the SQL query. The number of characters for an SQL query displayed in the SQL Statement column is based on the number defined in Preferences > Editor > SQL History pane. Refer to the SQL History section to modify the number of characters for a query.

● **Number of Records** - Displays the amount of records fetched by the SQL query.
● **Start Time** - Displays the time the query execution was started.
● **Execution Time** - Displays the time taken to execute the query.
● **Database Name** - Displays the name of the database.
● **Execution Status** - Displays the execution status of the query as **Success** or **Failure**.

Deleting the connection profile deletes the history. If the **SQL History** dialog box is closed, the query is not removed from the list.

You can perform the following operations in the **SQL History** dialog box:

- Loading an SQL query into the SQL terminal
- Loading multiple SQL queries into the SQL terminal
- Deleting an SQL query
- Deleting all SQL queries
- Pinning an SQL query
- Unpinning an SQL query

### Loading an SQL query into the SQL terminal

Follow the steps to load the SQL query into the SQL terminal:

**Step 1** Select the required query and click ![ ].

The query is appended to the cursor position in the **SQL Terminal**.

--- End

### Loading multiple SQL queries into the SQL terminal

The **Load in SQL Terminal and close History** button loads selected queries into the **SQL Terminal** and closes the **SQL History** dialog box.

Follow the steps to load selected SQL queries into the SQL terminal:

**Step 1** Select the required queries.

**Step 2** Click ![ ].

The queries are appended to the cursor position in the **SQL Terminal**.

--- End

**NOTE**

If you continue the execution on error, then each statement in the terminal will be running as a scheduled job and runs one after the other. The execution status is updated in the console and job is listed in the progress bar. When the time difference between Job Execution, Progress Bar Update and Console Update is very minimal, you will not be able to open the progress bar and stop the execution. In such scenarios you have to close the SQL Terminal to come out of execution.

### Deleting an SQL query

Follow the steps to delete a SQL query from the SQL History list:
Step 1 Select the required query and click 🖼.
A confirmation pop up window is displayed.
Step 2 Click OK.

---End

Deleting all SQL queries

Follow the steps to delete all SQL queries from the SQL History list:

Step 1 Click 🖼.
A confirmation pop up window is displayed.
Step 2 Click OK.

---End

Pinning an SQL query

You can pin queries that you do not want Data Studio to delete automatically from the SQL History. You can pin a maximum of 50 queries. Pinned queries are displayed at the top of the list. The value set in SQL History count does not affect the pinned queries. Refer to SQL History section for additional information on SQL History count.

📖 NOTE

The pinned queries appear on top once the SQL History window is closed and re-opened.

Follow the steps to pin a SQL query:

Step 1 Select the required SQL query and click 🖼.
The Pin Status column displays the pinned status of the query.

----End

Unpinning an SQL query

Follow the steps to unpin a SQL query:

Step 1 Select the required SQL query and click 🖼.
The Pin Status column displays the unpinned status of the query.

----End

4.21.3 Opening And Saving SQL Scripts

Opening an SQL Script

Follow the steps to open an SQL script:
**Step 1** Choose File > Open from the main menu. Alternatively, click Open on the toolbar or right-click the SQL Terminal and select Open.

If the SQL Terminal has existing content, then there will be an option to override the existing content or append content to it.

**Step 2** The Open dialog box is displayed.

**Step 3** In Open dialog box, select the SQL file to import and click Open.

The selected SQL script is opened as a File Terminal.

Icon is different. On mouse over the source file and corresponding database connection will be displayed on File Terminal.

----End

**NOTE**

- The encoding type of the SQL file must match the encoding type specified in preferences.
- Label of the file terminal will start with * if any of its content is edited. Dirty flag is removed once the file terminal is saved.
- File Terminals cannot be renamed, one terminal is always mapped to one Source Script File, but one script can be opened in multiple terminals.
- You can open SQL scripts only on SQL Terminals.

Data Studio allows you to save and open SQL scripts in the SQL Terminal. After saving the changes, SQL Terminal will be changed to a File Terminal.

**Saving an SQL Script**

Save option saves the File Terminal content to the associated file.

Follow the steps to save an SQL script:

**Step 1** Perform any of the following operations:

- Choose File > Save from the main menu.
- Press "Ctrl + s" to save the SQL terminal content.
- Click Save on the toolbar or right-click the SQL Terminal and select Save.

The Data Studio Security Disclaimer dialog box is displayed.

**Step 2** Click OK.

Data Studio displays the status of the operation in the status bar.
The script is saved as an SQL file. Data Studio sets the read/write permission for the saved SQL file. To ensure security, you must set the read/write permissions for folders.

When a change is made in a file and if that associated file is unavailable, it will trigger Save As option.
- In any case, if saving of the source file is failed due to some reason, then user is prompted with Save As option to save the content as a new source file. If you choose not to save (that is cancel Save As), then File Terminal gets converted into an SQL Terminal.
- The changes made to File Terminals are not Auto Saved.

---End

**Saving an SQL Script in New File**

Save As option saves the terminal content to a new file.

Follow the steps to save an SQL script:

**Step 1** Perform any of the following operations:
- Choose File > Save As from the main menu.
- Alternatively click "ctrl +Alt+ s" key to save SQL Terminal or File Terminal content in new file.

The Data Studio Security Disclaimer dialog box is displayed.

**Step 2** Click OK.

The Save As dialog box is displayed.

**Step 3** Select the location to save the script and click Save.

---End

**NOTE**

When there are unsaved changes in File Terminals, then user will be given an option to save or cancel on graceful exit of data studio.

### 4.21.4 Viewing Object Properties in the SQL Terminal

Data Studio allows you to view table properties and functions/procedures.

Follow the steps to view table properties:

**Step 1** Press Ctrl and point to the table name.

**Step 2** Click the highlighted table name. The properties of the selected table is displayed.
**NOTE**

The table properties are read-only.

-----End

Follow the steps to view functions/procedures:

**Step 1** Press Ctrl and point to the function/procedure name.

```
CREATE OR REPLACE FUNCTION hello_hello()
    RETURNS void
    LANGUAGE plpgsql
    AS $`
```

**Step 2** Click the highlighted function/procedure name. The function/procedure is displayed in a new **PL/SQL Viewer** tab.

-----End

Follow the steps to view the properties of a View:

**Step 1** Press Ctrl and point to the view name.

**Step 2** Click the highlighted view name. The properties of the selected view is displayed.

-----End

**Saving a Terminal Content Before Exiting Application**

Data Studio allows you to save the unsaved content of the terminal before exiting the application.

Follow the steps to save the content of the terminal:

**Step 1** Click on close button of the application. **Exit Application** dialog box is displayed.

**Step 2** Click **Graceful Exit**.

1. **Saving File Terminal** dialog box appears. Unsaved dirty file terminal is displayed.

**Step 3** Select the terminal to save.

**Step 4** Click OK.

-----End

**NOTE**

Saving File Terminal dialog box does not appear in case of Force Exit.
4.21.5 Canceling Execution of SQL Queries

Data Studio allows you to cancel execution of an SQL query executing in the SQL Terminal.

Follow the steps to cancel execution of an SQL query:

**Step 1** Execute the SQL query in the SQL Terminal.

**Step 2** Click in the SQL Terminal or press Shift+Esc.

Alternatively, you can choose Run > Cancel from the main menu or right-click in the SQL Terminal and select Cancel, or from Progress View tab select Cancel.

When you cancel the query, the execution stops at the currently executing SQL statement.

Database changes made by the canceled query are rolled back and the queries following the canceled query are not executed.

A query is not canceled and the Result tab shows the result when:

1. The server has finished execution of the query and is preparing the result.
2. The result of the executed query is being transferred from the server to the Data Studio client.

A query cannot be canceled while viewing the query Execution Plan. For more details, refer to Viewing the Query Execution Plan and Cost.

The Messages tab shows the query cancelation message.

--- END

4.21.6 Formatting of SQL Queries

Data Studio supports formatting and highlighting of SQL queries and PL/SQL statements.

**PL/SQL Formatting**

Follow the steps to format PL/SQL statements:

**Step 1** Select the PL/SQL statement to be formatted.

**Step 2** Click on the toolbar to format the query.

Alternatively, use Ctrl+Shift+F or choose Edit > Format from the main menu.

The PL/SQL statements are formatted.
SQL Formatting

Data Studio supports formatting of simple SQL SELECT, INSERT, UPDATE, DELETE statements which are syntactically correct. Following are some of the statements for which formatting is supported.

1. The SELECT statement must be made of following clauses:
   - Target list
   - From clause (includes join)
   - Where clause
   - Group by clause
   - Having clause
   - Order by clause
   - Common table expression

   SELECT statement without SET operations such as UNION, UNION ALL, MINUS, INTERSECT and so on.

   SELECT statements without sub-queries.

2. The INSERT statement is made of below clauses only
   - Insert Into Table name
   - Values clause
   - Values Column List
   - RETURNING

3. The UPDATE statement is made of below clauses only
   - Update Table name
   - SET Clause
   - From Clause (Including Join)
   - Where Clause
   - RETURNING

4. The DELETE statement is made of below clauses only
   - Delete From Table name
   - Using Clause (Including Join)
   - Where Clause
   - RETURNING

Follow the steps to format SQL queries:

**Step 1** Select the SQL query statements to be formatted.

**Step 2** Click on the toolbar to format the query.

Alternatively, use the Ctrl+Shift+F or choose Edit > Format from the main menu.

The query is formatted.

Refer following table for query formatting rules:
<table>
<thead>
<tr>
<th>Statement</th>
<th>Clauses</th>
<th>Format Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT</td>
<td>SELECT list</td>
<td>Line break before first column&lt;br&gt;Indent column list</td>
</tr>
<tr>
<td>FROM</td>
<td>Line break before FROM&lt;br&gt;Line break after FROM&lt;br&gt;Indent FROM list&lt;br&gt;Stack FROM list</td>
<td></td>
</tr>
<tr>
<td>JOIN (FROM clause)</td>
<td>Line break before JOIN&lt;br&gt;Line break after JOIN&lt;br&gt;Line break before ON&lt;br&gt;Line break after ON&lt;br&gt;Indent table after JOIN&lt;br&gt;Indent ON condition</td>
<td></td>
</tr>
<tr>
<td>WHERE</td>
<td>Line break before WHERE&lt;br&gt;Line break after WHERE&lt;br&gt;Indent WHERE condition&lt;br&gt;Place WHERE condition on single line</td>
<td></td>
</tr>
<tr>
<td>GROUP BY</td>
<td>Line break before GROUP&lt;br&gt;Line break before GROUP BY expression&lt;br&gt;Indent column list&lt;br&gt;Stack column list</td>
<td></td>
</tr>
<tr>
<td>HAVING</td>
<td>Line break before HAVING&lt;br&gt;Line break after HAVING&lt;br&gt;Indent HAVING condition</td>
<td></td>
</tr>
<tr>
<td>ORDER BY</td>
<td>Line break before ORDER&lt;br&gt;Line break after BY&lt;br&gt;Indent column list&lt;br&gt;Stack column list</td>
<td></td>
</tr>
<tr>
<td>CTE</td>
<td>Indent subquery braces&lt;br&gt;Each CTE in a new line</td>
<td></td>
</tr>
<tr>
<td>Statement</td>
<td>Clauses</td>
<td>Format Rules</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INSERT</td>
<td>INSERT INFO</td>
<td>Line break before opening brace</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Line break after opening brace</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Line break before closing brace</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indent column list braces</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indent column list</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Line break before VALUES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stack column list</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Line break before VALUES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Line break before opening brace</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Line break after opening brace</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Line break before closing brace</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indent VALUES expressions list braces</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indent VALUES expressions list</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stack VALUES expressions list</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Line break before DEFAULT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indent DEFAULT keyword</td>
</tr>
<tr>
<td>DEFAULT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTE</td>
<td></td>
<td>Each CTE in a new line</td>
</tr>
<tr>
<td>RETURNING</td>
<td></td>
<td>Line break before RETURNING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Line break after RETURNING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indent RETURNING column list</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Place RETURNING column List on single line</td>
</tr>
<tr>
<td>UPDATE</td>
<td>UPDATE Table</td>
<td>Line break before table</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indent table</td>
</tr>
<tr>
<td>SET Clause</td>
<td></td>
<td>Line break before SET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indent column assignments list</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indent column assignments list</td>
</tr>
<tr>
<td>FROM CLAUSE</td>
<td></td>
<td>Line break before FROM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Line break after FROM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indent FROM list</td>
</tr>
<tr>
<td>Statement</td>
<td>Clauses</td>
<td>Format Rules</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stack FROM list</td>
</tr>
<tr>
<td>JOIN CLAUSE(FROM CLAUSE)</td>
<td>Line break before JOIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Line break after JOIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Line break before ON</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Line break after ON</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indent table after JOIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indent ON condition</td>
</tr>
<tr>
<td>WHERE CLAUSE</td>
<td>Line break before WHERE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Line break after WHERE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indent WHERE condition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indent WHERE condition</td>
</tr>
<tr>
<td>CTE</td>
<td></td>
<td>Each CTE in a new line</td>
</tr>
<tr>
<td>RETURNING</td>
<td>Line break before RETURNING</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Line break after RETURNING</td>
</tr>
<tr>
<td>DELETE</td>
<td></td>
<td>Indent RETURNING column list</td>
</tr>
<tr>
<td>USING CLAUSE</td>
<td>Line break before FROM</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Line break after FROM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indent USING list</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stack FROM list</td>
</tr>
<tr>
<td>JOIN CLAUSE</td>
<td>Line break before JOIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Line break after JOIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Line break before ON</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Line break after ON</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indent table after JOIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indent ON condition List</td>
</tr>
<tr>
<td>WHERE CLAUSE</td>
<td>Line break before WHERE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Line break after WHERE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indent WHERE condition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stack WHERE condition list</td>
</tr>
<tr>
<td>Statement</td>
<td>Clauses</td>
<td>Format Rules</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>--------------</td>
</tr>
<tr>
<td>CTE</td>
<td>Each CTE in a new line</td>
<td></td>
</tr>
<tr>
<td>RETURNING</td>
<td>Line break before RETURNING</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Line break after RETURNING</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Indent RETURNING column list</td>
<td></td>
</tr>
</tbody>
</table>

----End

Data Studio supports automatic highlighting of following punctuation mark's pair when cursor is placed before or after the punctuation mark or selection of the punctuation mark.

- Brackets - ( )
- Square brackets - [ ]
- Braces - { }
- Single-quoted string literals - ' '
- Double-quoted string literals - " "

Follow the steps below to change case for SQL queries and PL/SQL statements:

Text case can be changed in the SQL Terminal using one of the following methods:

**Method 1:**
**Step 1** Select the text, choose Edit > Upper Case/Lower Case.
The text changes to the case selected.

----End

**Method 2:**
**Step 1** Select the text, choose AA or BB from toolbar.
The text changes to the case selected.

----End

**Method 3:**
**Step 1** Select the text, press Ctrl+Shift+U to change to upper case or Ctrl+Shift+L to change to lower case.
The text changes to the case selected.

----End

**SQL Highlight**

Keywords are highlighted automatically as you enter them (according to the default color scheme) as shown below:
The following figure shows the default color scheme for the specified type of syntax:

<table>
<thead>
<tr>
<th>Syntax Category</th>
<th>Color Value</th>
<th>HEX Value</th>
<th>Shade</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>RGBA(0,0,0)</td>
<td>000000</td>
<td></td>
</tr>
<tr>
<td>UNRESERVED_KEYWORD</td>
<td>RGBA(198,134)</td>
<td>C60086</td>
<td></td>
</tr>
<tr>
<td>TYPE</td>
<td>RGBA(64,200)</td>
<td>4000C8</td>
<td></td>
</tr>
<tr>
<td>PREDICATES</td>
<td>RGBA(224,11)</td>
<td>E000B</td>
<td></td>
</tr>
<tr>
<td>RESERVED</td>
<td>RGBA(15,80)</td>
<td>235A50</td>
<td></td>
</tr>
<tr>
<td>CONSTANTS</td>
<td>RGBA(15,80)</td>
<td>235A50</td>
<td></td>
</tr>
<tr>
<td>SINGLE_LINE_COMMENT</td>
<td>RGBA(64,128)</td>
<td>4000B8</td>
<td></td>
</tr>
<tr>
<td>STRING</td>
<td>RGBA(0,255)</td>
<td>0000FF</td>
<td></td>
</tr>
</tbody>
</table>

Refer to Syntax Coloring to customize the SQL highlight color scheme for the specific type of syntax.

### 4.21.7 Selecting a DB Object in the SQL Terminal

Data Studio suggests a list of possible schema names, table names and column names, and views in the **SQL Terminal**.

Follow the steps to select a DB object:

**Step 1** Press Ctrl+Space and enter the required parent DB object name. The DB objects list is refined as you continue typing the DB object name. The DB objects list displays all DB objects of the database connected to the **SQL Terminal**.

**Step 2** To select the parent DB object, use the **Up** or **Down** arrow keys and press **Enter** on the keyboard, or double-click the parent DB object.
Step 3  Enter . (period) to list all child DB objects.

![Image of SQL query]

Step 4  To select the child DB object, use the Up or Down arrow keys and press Enter on the keyboard, or double-click the child DB object.

On selection, the child DB object will be appended to the parent DB object (with a period '.').

**NOTE**

- Auto-suggest also works on schema names, table names, views, and table name aliases in the same way as shown above for all schema objects that you have access. Auto-suggest for column name alias is not supported.

Following is a sample query with alias objects:

```sql
SELECT table_alias.<auto-suggest>
FROM test.t1 AS table_alias
WHERE table_alias.<auto-suggest> = 5
GROUP BY table_alias.<auto-suggest>
HAVING table_alias.<auto-suggest> = 5
ORDER BY table alias.<auto-suggest>
```

- Auto-suggest may show "Loading" in Terminal for following scenarios:
  - The object is not loaded due to the value mentioned in the **Load Limit** field. Refer to Adding a Connection for more information.
  - The object is not loaded since it is added in the **Exclude** list option. Refer to Adding a Connection for more information.
  - There is a delay in fetching the object from the server.

- If there are objects with the same name in different case, then auto-suggest will display child objects of both parent objects.

**Example:**
If there are two schemas with the name public and PUBLIC, then all child objects for both these schemas will be displayed.

---End

4.21.8 Viewing the Query Execution Plan and Cost

The execution plan shows how the table(s) referenced by the SQL statement will be scanned (plain sequential scan and index scan).
The SQL statement execution cost is the estimate at how long it will take to run the statement (measured in cost units that are arbitrary, but conventionally mean disk page fetches).

Follow the steps to view the plan and cost for a required SQL query:

**Step 1** Enter the query or use an existing query in the **SQL Terminal** and click on the SQL Terminal toolbar to view explain plan.

To view explain analyze, click the drop-down from , select **Include Analyze**, and click .

The **Execution Plan** opens in tree view format as a new tab at the bottom by default. The display mode has a tree shape and text style.

**NOTE**

The data shown in tree explain plan and visual explain may vary, since the execution parameters considered by both are not the same.

Following are the parameters selected for explain plan with/without analyze and the columns displayed:

**Table 4-19** Explain Plan Options

<table>
<thead>
<tr>
<th>Explain Plan Type</th>
<th>Parameters Selected</th>
<th>Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Include Analyze unchecked (default setting)</td>
<td>Verbose, Costs</td>
<td>Node type, startup cost, total cost, rows, width, and additional Info</td>
</tr>
<tr>
<td>Include Analyze checked</td>
<td>Analyze, Verbose, Costs, Buffers, Timing</td>
<td>Node type, startup cost, total cost, rows, width, Actual startup time, Actual total time, Actual Rows, Actual loops, and Additional Info</td>
</tr>
</tbody>
</table>

Additional Info column includes, predicate information (filter predicate, hash condition), distribution key and output information along with the node type information.

The tree view of plan categorizes nodes into 16 types. In tree view, each node will be preceded with corresponding type of icon. Following is the list of node categories with icons:

**Table 4-20** Node Categories with Icon

<table>
<thead>
<tr>
<th>Node Category</th>
<th>Icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregate</td>
<td>🎊</td>
</tr>
<tr>
<td>Node Category</td>
<td>Icon</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------</td>
</tr>
<tr>
<td>Group Aggregate</td>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>Function</td>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>Hash</td>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>Hash Join</td>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>Nested Loop</td>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>Nested Loop Join</td>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>Modify Table</td>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>Partition Iterator</td>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>Row Adapter</td>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>Seq Scan on</td>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>Set Operator</td>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>Sort</td>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>Stream</td>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>Union</td>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>Unknown</td>
<td><img src="image" alt="Icon" /></td>
</tr>
</tbody>
</table>

Hover over the highlighted cells to identify the heaviest, costliest, and slowest node. Cells will be highlighted only for tree view.

If multiple queries are selected, explain plan with/without analyze will be displayed only for last query selected.

Each time execution plan is executed, the plan opens in a new tab.

If the connection is lost and the database is still connected in Object Browser, then **Connection Error** dialog box is displayed:

- **Yes** - The connection is reestablished and retrieves explain plan and cost.
- **No** - Disconnects database in Object Browser.
Toolbar menu in the **Execution Plan** window:

<table>
<thead>
<tr>
<th>Toolbar Name</th>
<th>Toolbar Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree Format</td>
<td>![Tree Format Icon]</td>
<td>This icon is used view explain plan in tree format.</td>
</tr>
<tr>
<td>Text Format</td>
<td>![Text Format Icon]</td>
<td>This icon is used view explain plan in text format.</td>
</tr>
<tr>
<td>Copy</td>
<td>![Copy Icon]</td>
<td>This icon is used to copy selected content from result window to clipboard. Shortcut key - <strong>Ctrl+C</strong>.</td>
</tr>
<tr>
<td>Save</td>
<td>![Save Icon]</td>
<td>This icon is used to save the explain plan in text format.</td>
</tr>
</tbody>
</table>

Refer to **Execute SQL Queries** for information refresh, SQL preview, and search bar.

Refresh operation re-executes the explain/analyze query and refreshes the plan in the existing tab.

The result is displayed in the **Messages** tab.

---

**4.21.9 Viewing the Query Execution Plan and Cost Graphically**

Visual Explain plan displays a graphical representation of the sql query using information from the extended JSON format. This helps to refine query to enhance query and server performance. It helps to analyze the query path taken by the database and identifies heaviest, costliest and slowest node.

The graphical execution plan shows how the table(s) referenced by the SQL statement will be scanned (plain sequential scan and index scan).

The SQL statement execution cost is the estimate at how long it will take to run the statement (measured in cost units that are arbitrary, but conventionally mean disk page fetches).

**Costliest** - Highest **Self Cost** plan node.

**Heaviest** - Maximum number of rows output by a plan node is considered heaviest node.

**Slowest** - Highest execution time by a plan node.

Follow the steps to view the graphical representation of plan and cost for a required SQL query:

**Step 1** Enter the query or use an existing query in the **SQL Terminal** and click ![Visual Plan Analysis Icon] on the SQL Terminal toolbar. Alternatively, press ALT+CTRL+X together.

**Visual Plan Analysis** window is displayed.
Refer to **Viewing the Query Execution Plan and Cost** section for information on reconnect option in case connection is lost while retrieving the execution plan and cost.

- **1 - General Detail tab** - This tab displays the query.
- **2 - Visual Explain Plan tab** - This tab displays a graphical representation of all nodes like execution time, costliest, heaviest, and slowest node. Click each node to view the node details.
- **3 - Properties - General tab** - Provides the execution time of the query in ms.
- **4 - Properties - All Nodes tab** - Provides all node information.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node Name</td>
<td>Name of the node</td>
</tr>
<tr>
<td>Analysis</td>
<td>Node analysis information is provided.</td>
</tr>
<tr>
<td>RowsOutput</td>
<td>Number of rows output by the plan node</td>
</tr>
<tr>
<td>RowsOutput Deviation (%)</td>
<td>Deviation % between estimated rows output and actual rows output by the plan node</td>
</tr>
<tr>
<td>Execution Time (ms)</td>
<td>Execution time taken by the plan node</td>
</tr>
<tr>
<td>Contribution (%)</td>
<td>Percentage of the execution time taken by plan node against the overall query execution time.</td>
</tr>
<tr>
<td>Self Cost</td>
<td><strong>Total Cost</strong> of the plan node - <strong>Total Cost</strong> of all child nodes</td>
</tr>
<tr>
<td>Total Cost</td>
<td>Total cost of the plan node</td>
</tr>
</tbody>
</table>

- **5 - Properties - Exec. Plan tab** - Provides the execution information of all nodes.
<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node Name</td>
<td>Name of the node</td>
</tr>
<tr>
<td>Entity Name</td>
<td>Name of the object</td>
</tr>
<tr>
<td>Cost</td>
<td>Execution time taken by the plan node</td>
</tr>
<tr>
<td>Rows</td>
<td>Number of rows output by the plan node</td>
</tr>
<tr>
<td>Loops</td>
<td>Number of loops of execution performed by each node.</td>
</tr>
<tr>
<td>Width</td>
<td>The estimated average width of rows output by the plan node in bytes</td>
</tr>
<tr>
<td>Actual Rows</td>
<td>Number of estimated rows output by the plan node</td>
</tr>
<tr>
<td>Actual Time</td>
<td>Actual execution time taken by the plan node</td>
</tr>
</tbody>
</table>

- **6 - Plan Node - General** tab - Provides the node information for each node.

<table>
<thead>
<tr>
<th>Row Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>Provides the column information returned by the plan node</td>
</tr>
<tr>
<td>Analysis</td>
<td>Provides analysis of the plan node like costliest, slowest, and heaviest.</td>
</tr>
<tr>
<td>RowsOutput Deviation (%)</td>
<td>Deviation % between estimated rows output and actual rows output by the plan node</td>
</tr>
<tr>
<td>Row Width (bytes)</td>
<td>The estimated average width of rows output by the plan node in bytes</td>
</tr>
<tr>
<td>Plan Output Rows</td>
<td>Number of rows output by the plan node</td>
</tr>
<tr>
<td>Actual Output Rows</td>
<td>Number of estimated rows output by the plan node</td>
</tr>
<tr>
<td>Actual Startup Time</td>
<td>The actual execution time taken by the plan node to output the first record</td>
</tr>
<tr>
<td>Actual Total Time</td>
<td>Actual execution time taken by the plan node</td>
</tr>
<tr>
<td>Row Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Startup Cost</td>
<td>The execution time taken by the plan node to output the first record</td>
</tr>
<tr>
<td>Total Cost</td>
<td>Execution time taken by the plan node</td>
</tr>
<tr>
<td>Is Column Store</td>
<td>This field represents the orientation of the table (column or row store)</td>
</tr>
<tr>
<td>Node Type</td>
<td>Represents the type of node</td>
</tr>
<tr>
<td>Parent Relationship</td>
<td>Represents the relationship with the parent node</td>
</tr>
</tbody>
</table>

Based on the plan node type additional information may display. Few examples:

<table>
<thead>
<tr>
<th>Plan Node</th>
<th>Additional Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitioned CStore Scan</td>
<td>Table Name, Table Alias, Schema Name</td>
</tr>
<tr>
<td>Vector Sort</td>
<td>Sort keys</td>
</tr>
<tr>
<td>Vector Hash Aggregate</td>
<td>Group By Key</td>
</tr>
<tr>
<td>Vector Has Join</td>
<td>Join Type, Hash Condition</td>
</tr>
<tr>
<td>Vector Streaming</td>
<td>Distribution key, Spawn On</td>
</tr>
</tbody>
</table>

- **7 - Plan Node - DN Details** tab - Provides detailed data node information for each node. **Data Node** (DN) details are available only if data is being collected from data node.

  Refer to **Viewing Table Data** section for description on copy and search toolbar options.

--- End

**4.21.10 Working with the SQL Terminals**

In the **SQL Terminal**, you can

- **Auto Commit**
- **Execute SQL Queries**
- **Backup Unsaved Queries/Functions/Procedures**
- **Locate Error**
- **Search in PL/SQL Viewer or SQL Terminal**
- **Go to Line in PL/SQL Viewer or SQL Terminal**
- **Comment/Uncomment**
Auto Commit

Auto Commit option can be switch on or off based on the Preferences settings. Refer Transaction section for further details on how to enable and disable Auto Commit option.

- If Auto Commit option is enabled, Commit and Rollback buttons are disabled. Transactions are committed automatically.
- If Auto Commit option is disabled, Commit and Rollback buttons are enabled. You can use the buttons manually to commit or revert the changes.

**NOTE**
- For OLAP, server opens a transaction for all the SQL statements. (For Example: select statement, explain select statement, set parameter)
- For OLTP, server opens transaction for only DML statements. (For Example: INSERT statement, UPDATE statement, DELETE statement)

Reuse Connection

It enables the user to choose the same SQL terminal connection or new connection for the result set. The choice affects the record visibility as per the isolation levels defined in the database server.

- When Reuse Connection is ON, terminal connection will be used for data manipulation and refresh of the result window.

For some database temp tables that are created or used by the terminal can be edited from the result window.

- When Reuse Connection is OFF, new connection will be used for data manipulation and refresh of the result window.

For some database temp tables cannot be edited from the result window.

Icon is displayed when Reuse Connection is ON.

Icon is displayed when Reuse Connection is OFF.

Icon is displayed when Reuse Connection is disabled.

**Follow the steps to turn off Reuse Connection:**

**Step 1** Click on the SQL Terminal toolbar.

Reuse Connection is disabled for the terminal.
**NOTE**

- Auto Commit ON - Reuse Connection is enabled and ON by default. You can switch it OFF if needed.
- Auto Commit OFF - Reuse Connection is disabled and ON.

----End

Refer to Table 4-24 for more details about the behavior of Auto Commit and Reuse Connection.

**Execute SQL Queries**

Follow the steps to execute function/procedure(s) or SQL queries

Enter a function/procedure(s) or SQL query(s) in the SQL Terminal tab and click in the SQL Terminal tab, or press Ctrl+Enter, or choose Run > Compile/Execute Statement from the main menu.

Alternatively, you can right-click in the SQL Terminal tab and select Execute Statement.

**NOTE**

You can check the status bar to view the status of a query being executed.

The Result tab displays the results after executing the function/procedure(s) or SQL queries along with the query statement executed. If the connection is lost during execution and the database is still connected in Object Browser, then Connection Error dialog box is displayed:

- **Reconnect** - The connection is reestablished.
- **Reconnect and Execute** - With Auto commit on, execution will continue from failure statement. With Auto commit off, execution will continue from position of cursor.
- **Cancel** - Disconnects database in Object Browser.

Failure to reconnect after three attempts will disconnect the database in Object Browser. Connect to the database in Object Browser and retry execution.

**NOTE**

- For long running queries, result set can be edited only after the complete results are fetched.
- Editing of query results are only allowed in following scenarios:
  - Select is from a single table
  - Either select all columns or subset of columns [No aliases, aggregate functions, expressions on columns]
  - All WHERE condition
  - All ORDER BY clause
  - On regular, partition, and temporary tables.
- Committing an empty row assigns Null to all columns.
- Only result set of queries executed on tables available in Object Browser is editable.
- Editing of query results is allowed only for queries executed in SQL Terminal.
The column width definition can be set using **Settings > Preferences** option. Refer to **Query Results** to set this parameter.

**Column Reorder**

Column reordering can be performed by clicking and dragging the selected column header to the desired position.

**Multi-Column Sort**

This feature allows users to sort table data of some pages by multiple columns. In addition, you can set the priority of columns for sorting.

The feature is available for the following pages:

- Result Set Tab
- Edit Table Data Window
- View Table Data Window
- Batch Drop Result Window

Follow the steps to access Multi-column sort:

**Step 1** Click button on the toolbar.

**Multi-Column Sort** pop-up is displayed.

**Step 2** Click **Add Column**. Choose the column you want to sort from the drop down.


**Step 3** Select the required sort order.

**Step 4** Click on Apply.

---

Multi-sort pop up has following elements:

**Table 4-21 Elements Of Multi-Column Pop-up:**

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>User Interface (UI) Element Type</th>
<th>Description/Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>Read only text field</td>
<td>Shows column priority in multi sort.</td>
</tr>
<tr>
<td>Column Name</td>
<td>Combo field having all column names of the table as its value set</td>
<td>Column name of the column added for sorting.</td>
</tr>
<tr>
<td>Data Type</td>
<td>Read only text field</td>
<td>Shows data type of the column selected.</td>
</tr>
<tr>
<td>Sort Order</td>
<td>Combo field having values {sort_ascending, sort_descending}</td>
<td>Sort order of the column.</td>
</tr>
<tr>
<td>Add Column</td>
<td>Button</td>
<td>Adds new row to multi-sort table.</td>
</tr>
<tr>
<td>Delete Column</td>
<td>Button</td>
<td>Deletes selected column from multi-sort table.</td>
</tr>
<tr>
<td>Up</td>
<td>Button</td>
<td>Moves selected column up by 1 step, thus changing sort priority.</td>
</tr>
<tr>
<td>Down</td>
<td>Button</td>
<td>Moves selected column down by 1 step, this changing sort priority.</td>
</tr>
<tr>
<td>Apply</td>
<td>Button</td>
<td>Apply prepared sort configuration.</td>
</tr>
</tbody>
</table>
**NOTE**

Except following data types, all the other data types will be sorted by their string value (Alphabetical order):

- TINYINT
- SMALLINT
- INTEGER
- BIGINT
- FLOAT
- REAL
- DOUBLE
- NUMERIC
- BIT
- BOOLEAN
- DATE
- TIME
- TIME_WITH_TIMEZONE
- TIMESTAMP
- TIMESTAMP_WITH_TIMEZONE

Following icons are provided in Multi-Column sort:

Elements of Multi-Column Pop-up:

**Table 4-22** Icons Of Multi-Column Sort

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Not Sorted" /></td>
<td>Not Sorted</td>
<td>This icon in column header indicates that the column is not sorted. If you click on this icon the column will be sorted in ascending order. Alternatively click Alt +column header</td>
</tr>
<tr>
<td><img src="image" alt="Ascending Sort" /></td>
<td>Ascending Sort</td>
<td>This icon in column header indicates that the column is sorted in ascending order. If you click on this icon, the column will be sorted in descending order. Alternatively click Alt +column header</td>
</tr>
<tr>
<td>Icon</td>
<td>Description</td>
<td>Action</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>--------</td>
</tr>
<tr>
<td><img src="image" alt="Descending Sort" /></td>
<td>Descending Sort</td>
<td>This icon in column header indicates that the column is sorted in descending order. If you click on this icon the column will be in no sort order. Alternatively click $\text{Alt}+\text{column}$ header</td>
</tr>
</tbody>
</table>

Icons for the sort priority are as following:

- ![Descending Sort](image): Icons having three dots have the highest priority.
- ![Descending Sort](image): Icons having two dots have the second highest priority.
- ![Descending Sort](image): Icons having three dots have the third and onwards priority.

**Table 4-23 Toolbar Menus**

<table>
<thead>
<tr>
<th>Toolbar Name</th>
<th>Toolbar Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy</td>
<td><img src="image" alt="Copy" /></td>
<td>This button is used to copy selected content from result window to clipboard. Shortcut key - $\text{Ctrl}+\text{C}$.</td>
</tr>
<tr>
<td>Advanced Copy</td>
<td><img src="image" alt="Copy" /></td>
<td>This button is used to copy content from result window to clipboard. Results can be copied to include column header. Refer to View Query Results to set this preference. Shortcut key - $\text{Ctrl}+\text{Shift}+\text{C}$.</td>
</tr>
</tbody>
</table>
| Export all data | ![Export](image) | This icon is used to export all data either in excel (xlsx/xls), CSV, Text or Binary format. Refer to Exporting Table Data. **NOTE**
- Columns mentioned in the query is auto-populated in the Selected Columns section with Available Columns section empty.
- To export the query results, the query is re-executed using a new connection. The exported results may differ from the data in the results tab.
- Disabled for explain/analyze queries. To export explain/analyze queries use the Export current page data option. |
<table>
<thead>
<tr>
<th>Toolbar Name</th>
<th>Toolbar Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Export current page data</td>
<td>![Excel Icon]</td>
<td>This button is used to export current page data to excel (xlsx/xls) or CSV.</td>
</tr>
<tr>
<td>Paste</td>
<td>![Paste Icon]</td>
<td>This button is used to paste copied information. Refer to <strong>Paste</strong> section for more information.</td>
</tr>
<tr>
<td>Add</td>
<td>![Add Icon]</td>
<td>This button is used to add a row to the result set. Refer to <strong>Insert</strong> section for more information.</td>
</tr>
<tr>
<td>Delete</td>
<td>![Delete Icon]</td>
<td>This button is used to delete a row from the result set. Refer to <strong>Delete</strong> section for more information.</td>
</tr>
<tr>
<td>Save</td>
<td>![Save Icon]</td>
<td>This button is used to save the changes made in the result set. Refer to <strong>Editing Table Data</strong> section for more information.</td>
</tr>
<tr>
<td>Rollback</td>
<td>![Rollback Icon]</td>
<td>This button is used to roll back the changes made to the result set. Refer to <strong>Editing Table Data</strong> section for more information.</td>
</tr>
<tr>
<td>Refresh</td>
<td>![Refresh Icon]</td>
<td>This button is used to refresh information in the result set. If multiple result sets are open for the same table, then changes made to one result set will reflect on the other post refresh. Similarly if the same table is edited, then the result set will be updated post refresh.</td>
</tr>
<tr>
<td>Clear Unique Key selection</td>
<td>![Clear Icon]</td>
<td>This button is used to clear the previous unique key selection. Refer to <strong>Editing Table Data</strong> section for more information.</td>
</tr>
<tr>
<td>Show/Hide Query bar</td>
<td>![Query Bar Icon]</td>
<td>This button is used to display/hide the query executed for that particular result set. This is a toggle button.</td>
</tr>
<tr>
<td>Show/Hide Search bar</td>
<td>![Search Bar Icon]</td>
<td>This button is used to display/hide the search text field. This is a toggle button.</td>
</tr>
</tbody>
</table>
| Encoding                     | ![Encoding Icon] | This field will be available based on the Preference > Result Management > Query Result > Result Advanced Copy settings. This drop-down is used to select the appropriate encoding to view the data accurately. The default encoding is UTF-8. Refer to **Result Data Encoding** section to set the encoding preference. **NOTE**  
  - Data editing except for data insertion is restricted once the default encoding is modified. |
| Multi Sort                   | ![Sort Icon] | This button brings up multi-sort pop up.                                                                  |
### Toolbar Icon Description

<table>
<thead>
<tr>
<th>Toolbar Name</th>
<th>Toolbar Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear Sort</td>
<td>![Clear Sort Icon]</td>
<td>This button is used to reset all the sorted column.</td>
</tr>
</tbody>
</table>

### Icons in Search field:

<table>
<thead>
<tr>
<th>Icon Name</th>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search</td>
<td>![Search Icon]</td>
<td>This icon is used to search the result set based on the criteria defined. Search text are case insensitive.</td>
</tr>
<tr>
<td>Clear Search Text</td>
<td>![Clear Search Text Icon]</td>
<td>This icon is used to clear the search text entered in the search field.</td>
</tr>
</tbody>
</table>

### Right-click options in the Result window:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close</td>
<td>Closes only the active result window.</td>
</tr>
<tr>
<td>Close Others</td>
<td>Closes all other result windows except for the active result window.</td>
</tr>
<tr>
<td>Close All</td>
<td>Closes all result windows including the active result window.</td>
</tr>
</tbody>
</table>

### Status information displayed in the Result window:

- **Query Submit Time** - Provides the query submitted time.
- Number of rows fetched with execution time is displayed. The default number of rows is displayed. If there are additional rows to be fetched, then it will be denoted with the word "more". You can scroll to the bottom of the table to fetch and display all rows.

### NOTICE

When viewing table data, Data Studio automatically adjusts the column widths for a good table view. Users can resize the columns as needed. If the text contents of a cell exceeds the total available display area, then resizing the cell column may cause DS to become unresponsive.
Each time a query is run in SQL Terminal tab, a new result window opens. To view the results in the new window, you must select the newly opened window.

Set the focusOnFirstResult configuration parameter to false to automatically set focus to the newly opened Result window. Refer to Installing and Configuring Data Studio for details.

Each row, column and selected cells can be copied from the result set.

Export all data operation will be successful event after the connection is removed.

If the content of the column have spaces between the words, then word wrap is applied to fit the column within the display area. Word wrap is not applied if the content does not have any spaces between the words.

Select part of cell content and press Ctrl+C or click to copy selected text from a cell.

The size of the column is determined by the maximum content length column.

You can save preference to define:
- Number of records to be fetched.
- Column width
- Copy option from result set.

Refer to Query Results for more information

If any column of resultset tab has Lock Image icon in it, then values are not editable.

Backup Unsaved Queries/Functions/Procedures

Data Studio creates back up of unsaved data in SQL Terminal and PL/SQL Viewer periodically based on the time interval defined in the Preferences tab. The data can be encrypted and saved based on Preference settings. Refer to Query/Function/Procedure Backup section to turn on/off backup, define time interval to save the data, and encrypt the saved data.

Unsaved changes of each SQL Terminal/PL/SQL Viewer are taken as backup and stored in DataStudio\UserData<user name>\Autosave folder. Backup files saved before unexpected shutdown of Data Studio will be available at next login.

In case there are unsaved data in SQL Terminal/PL/SQL Viewer, during graceful exit, Data Studio will wait for backup to complete before closing.

Error Locator

During execution of query/function/procedure in case of an error the error locator message is displayed.

Yes - Click Yes to continue with the execution.

No - Click No to stop the execution.

Select Do not show additional errors for this execution option to hide the error message popup from displaying while continuing with the current execution.

Line number and position of error displays in Messages tab. The corresponding line number is marked with icon along with red underline at the position of the error in the Terminal/PL/SQL Viewer. Hovering over displays the error.
message. Refer to FAQs section to understand in certain scenarios why the line number and error detail does not match.

NOTE

If the query/function/procedure is modified while execution is in progress, then error locator may not display the correct line and position number.

Search in PL/SQL Viewer or SQL Terminal

Follow the steps to search in PL/SQL Viewer or SQL Terminal:

F3 key is used to search next word and Shift+F3 key is used to search previous word. These shortcut keys will be enabled only after Ctrl+F is used to search a text. These keys will be active with the current search word until a new word is searched. The value searched using Ctrl+F and F3/Shift+F3 will be applicable only for the current instance.

Step 1 Choose Edit > Find and Replace from the main menu.

Alternatively press Ctrl+F.

Find and Replace dialog box is displayed.

Step 2 Enter the text to be searched in Find what field, and click the Find Next button.

The searched text is highlighted.

F3 and Shift+F3 key will now be enabled for forward and backward search.

NOTE

Select Wrap around option to continue the search after reaching the last line in the SQL queries or PL/SQL statements.

---End

Go to Line in PL/SQL Viewer or SQL Terminal

Go to line option is used to skip to a specific line in the terminal.

Follow the steps to go to a line in PL/SQL Viewer or SQL Terminal:

Step 1 Choose Edit > Go To Line from the main menu or

Alternatively press Ctrl+G.

Go to Line dialog box is displayed.

Step 2 Enter the desired number in the Enter the line number field, and then click the OK button.

The cursor moves to the beginning of the line entered in the Go to Line dialog box.
Below are invalid inputs to this field.

- Non-numeric value
- Special characters
- Line number entered does not exist in the editor.
- More than 10 digits is entered.

----End

Comment/Uncomment

Comment/uncomment option is used to comment/uncomment lines or block of lines.

Follow the steps to comment/uncomment lines in PL/SQL Viewer or SQL Terminal:

**Step 1** Select the lines to comment/uncomment.

**Step 2** Choose Edit option. Choose Comment/Uncomment Lines from the main menu, or alternatively press Ctrl+/ or right-click and select Comment/Uncomment Lines, the selected lines are commented/uncommented.

----End

Follow the steps to comment/uncomment block of lines/content in PL/SQL Viewer or SQL Terminal:

**Step 1** Select the lines/content to comment/uncomment.

**Step 2** Choose Edit option. Choose Comment/Uncomment Block from the main menu, or alternatively press Ctrl+Shift+/ or right-click and select Comment/Uncomment Block, the selected block of lines/content are commented/uncommented.

----End

Indent/Unindent Lines

The indent/un-indent option is used to shift lines according to the indent size defined in the Preferences tab.

Follow the steps to indent lines in PL/SQL Viewer or SQL Terminal:

**Step 1** Select the lines to indent.

**Step 2** Press Tab or click  
Shifts the selected line according to the indent size defined in the Preferences tab. Refer to Formatter section to modify the indent size.

----End

Follow the steps to un-indent lines in PL/SQL Viewer or SQL Terminal:

**Step 1** Select the lines to un-indent.
**Step 2** Press **Shift+Tab** or click ✂️.

Shifts the selected line according to the indent size defined in the **Preferences** tab. Refer to **Formatter** section to modify the indent size.

**NOTE**

Only selected lines that have available tab space will be un-indented. For example, if multiple lines are selected, and one of the selected line starts at position 1, then pressing **Shift+Tab** will un-indent all the lines except for the one starting at position 1.

----End

**Insert Space**

The **Insert Space** option is used to replace a tab with spaces based on the indent size defined in the **Preferences** tab.

Follow the steps below to replace a tab with spaces in PL/SQL Viewer or SQL Terminal:

**Step 1** Select the lines to replace tab with spaces.

**Step 2** Press **Tab** or **Shift+Tab**.

Replaces the tab with spaces according to the indent size defined in the **Preferences** tab. Refer to **Formatter** section to modify the indent size.

----End

**Execute Multiple Functions/Procedures or Queries**

Follow the steps to execute multiple functions/procedures:

Insert ‘/’ in a new line after the function/procedure in the **SQL Terminal**.

Add the new function/procedure in the next line.
Follow the steps to execute multiple SQL queries:

**Step 1** Enter multiple SQL queries in the **SQL Terminal** tab as follows:

```
select * from test;
select * from test;
select * from test;
select * from test;
select * from test;
select * from test;
```

**Step 2** Click in the **SQL Terminal** tab, or press Ctrl+Enter, or choose Run > Compile/Execute Statement from the main menu.

**NOTE**
- If the queries are not selected for execution, then only the query in the line where cursor is placed will be executed.
- If the cursor is placed next to an empty line, then the next available query statement will be executed.
- If the cursor is placed at the last line which is blank, then no query will be executed.
- If a single query is written in multiples lines and the cursor is placed at any line of the query, then that query is executed. Queries are separated using semicolon (;).

----End

Follow the steps to execute an SQL query after a function/procedure:
Insert ‘/’ in a new line after the function/procedure and click  in the SQL Terminal tab.

Follow the steps to execute PL/SQL statements and SQL queries on different connections:

In the toolbar, select the required connection from the connection profiles dropdown list and click  in the SQL Terminal tab.

**Rename SQL Terminal**

Follow the steps to rename SQL Terminal:

**Step 1** In the SQL Terminal tab right-click and select Rename Terminal.

A Rename Terminal dialog box is displayed prompting you to provide the new name for the Terminal.

**Step 2** Enter the new name and select OK to rename the Terminal.

📋 NOTE

- Terminal name follows Windows file naming convention.
- Rename Terminal accepts a maximum of 150 characters.
- Restore option is not available to revert to the default name. You must manually rename the Terminal to default name.
- Tool tip of the renamed Terminal will display the old name.

----End

**SQL Assistant**

The SQL Assistant tool provides suggestion or reference for the information entered in the SQL Terminal and PL/SQL Viewer. Follow the steps below to open SQL Assistant:

When Data Studio is launched SQL Assistant panel displays with related syntax topics. As you type a query in the SQL Terminal topics related to the query is displayed. It also provides precautions, examples, syntax, function, and parameter description. Select the text and use the right-click option to copy selected information or copy and paste to SQL Terminal.

📋 NOTE

- You can enable/disable the SQL Assistant tool permanently. Refer to 7.3-SQL Assistant to enable/disable this option.

- SQL Assistant icon ( ) from the toolbar can be used to open the SQL Assistant window.

**Using Templates**

Data Studio provides an option to insert frequently used SQL statements in the SQL Terminal/PL/SQL Viewer using the Templates option. Some of the commonly used SQL statements are saved for ease of use. You can create, modify existing templates or remove templates.
Refer to **Adding/Modifying Templates** section for information on adding, removing, and creating new templates.

The following table lists the default templates:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>df</td>
<td>delete from</td>
</tr>
<tr>
<td>is</td>
<td>insert into</td>
</tr>
<tr>
<td>o</td>
<td>order by</td>
</tr>
<tr>
<td>s*</td>
<td>select from</td>
</tr>
<tr>
<td>sc</td>
<td>select row count</td>
</tr>
<tr>
<td>sf</td>
<td>select from</td>
</tr>
<tr>
<td>sl</td>
<td>select</td>
</tr>
</tbody>
</table>

Follow the steps to use the **Templates** option:

**Step 1** Enter the name of the template in SQL Terminal/PL/SQL Viewer.

**Step 2** Press **Alt+Ctrl+Space**.

A list of saved template information is displayed. The list displayed is based on the following criteria.

<table>
<thead>
<tr>
<th>Exact Match</th>
<th>Display List</th>
</tr>
</thead>
</table>
| On          | Displays all entries that match the input text case.  
**Example:** Entering "SF" in SQL Terminal/PL/SQL Viewer displays all entries that start with "SF". |
| Off         | Displays all entries that match the input irrespective of the text case.  
**Example:** Entering "SF" in SQL Terminal/PL/SQL Viewer displays all entries that start with "SF", "Sf", "sF", or "sf". |

<table>
<thead>
<tr>
<th>Text Selection/Cursor Location</th>
<th>Display List</th>
</tr>
</thead>
<tbody>
<tr>
<td>A text is selected and the shortcut key is used</td>
<td>Displays entries that match the text before the selection to the nearest space or new line character.</td>
</tr>
<tr>
<td>Text Selection/Cursor Location</td>
<td>Display List</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>No text selected and the shortcut key is used</td>
<td>Displays entries that match the text before the cursor to the nearest space or new line character.</td>
</tr>
</tbody>
</table>

📝 NOTE
- Using the shortcut key without entering text in SQL Terminal/PL/SQL Viewer displays all entries in the Templates.
- If the text entered in SQL Terminal/PL/SQL Viewer has only a single match, then it will be replaced directly in the SQL Terminal/PL/SQL Viewer without listing them out.
- After you click Open SQL Assistant button, the SQL Assistant pane is displayed. If you close the pane, it will be closed for all the terminals unless you click Open SQL Assistant button again.

----End

4.21.11 Exporting Query Results

You can export the results of an SQL query into a CSV, Text or Binary file.

This section contains the following topics:
- Exporting all data
- Exporting current page data

Exporting all data

The following functions are disabled while the export operation is in progress:
- Executing SQL queries in the SQL Terminal
- Executing PL/SQL statements
- Debugging PL/SQL statements

Follow the steps below to export all results:

Step 1 Select the Result tab.

Step 2 Click Export ResultSet Data.

Export ResultSet Data window is displayed.

Refer to Exporting Table Data to complete the export operation.

📝 NOTE
You can check the status bar to view the status of the result being exported.

The Data Exported Successfully dialog box is displayed.

Step 3 Click OK. Data Studio displays the status of the operation in the Messages tab.
If the disk is full while exporting the results, then Data Studio displays an error in the Messages tab. Clean the disk, re-establish the connection and export the result data.

----End

The Messages tab shows the Execution Time, Total result records fetched, and the path where the file is saved.

Exporting current page data

It is recommended that export all results instead of exporting the current page. Follow the steps to export the current page:

Step 1 Select the Result tab.

Step 2 Click to export the current page. The Data Studio Security Disclaimer dialog box is displayed.

Step 3 Click OK.

Step 4 Select the location to save the current page.

You can check the status bar to view the status of the page being exported.

Step 5 Click Save. The Data Exported Successfully dialog box is displayed.

Step 6 Click OK. Data Studio displays the status of the operation in the Messages tab.

If the disk is full while exporting the results, then Data Studio displays an error in the Messages tab. Clean the disk, re-establish the connection and export the result data.

----End

4.21.12 Managing SQL Terminal Connections

Data Studio allows you to reuse an existing SQL Terminal connection or create a new SQL Terminal connection for execution plan and cost, visual explain plan, and operations in the resultset. By default, the SQL Terminal reuses the existing connection to perform these operations.

Use new connection when there are multiple queries queued for execution in existing connection as the queries are executed sequentially and there may be a delay. Always reuse existing connection while working on temp tables. Refer to the Editing Temporary Tables section to edit temp tables.

Complete the steps to enable or disable SQL Terminal connection reuse:

Step 1 Click to enable or disable SQL Terminal connection reuse.
Refer to the FAQs section for the behavior of query execution with reuse and new connection.

**NOTE**

Use the existing SQL Terminal connection to edit temporary tables.

---End

### 4.22 Batch Operation

#### 4.22.1 Overview

You can view database objects to which you have access in Object Browser in the tree format. For example, you can view the schema names within the selected database and the corresponding table names within the selected schema.

The **Object Browser** displays only the objects that satisfy the following minimum privilege type requirement for the current user.

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Privilege to Display in Object Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Connect</td>
</tr>
<tr>
<td>Schema</td>
<td>Usage</td>
</tr>
<tr>
<td>Table</td>
<td>Select</td>
</tr>
<tr>
<td>Column</td>
<td>Select</td>
</tr>
<tr>
<td>Sequences</td>
<td>Usage</td>
</tr>
<tr>
<td>Function/Procedure</td>
<td>Execute</td>
</tr>
<tr>
<td>Tablespace</td>
<td>Create</td>
</tr>
</tbody>
</table>

It is not necessary that child objects of a parent object to which you have access will be displayed in **Object Browser**. For example, if you have access to a table but not have access to one of the columns in that table, then **Object Browser** displays only the table with columns to which you have access. The columns which you do not have access are not displayed. If access to an object is revoked when an operation is being performed, then an error message is displayed stating that you do not have access to perform the operation, and the object is removed from **Object Browser** after refresh.

Following database objects are supported (displayed in the tree format):

- Schemas
- Functions/Procedures
- Tables
- Sequences
- Views
• Columns, Constraints, and Indexes
• Tables

All default created schemas except for public are grouped under Catalogs and user schemas are grouped under Schemas below the respective database.

4.22.2 Dropping Batch of Objects

The batch drop operation allows you select multiple objects to drop. You can also perform batch drop operation on searched objects.

**NOTE**

• Batch drop is allowed only within the database.
• Batch drop of system objects will result in error, since system objects cannot be dropped.

Follow the steps to drop objects in a batch.

**Step 1** Press **Ctrl+left-click** (select objects one by one) or **Shift+left-click** (select objects in a bunch) to select the objects to be dropped.

**Step 2** Right-click and select **Drop Objects**.

**Drop Objects** tab displays with the list of objects to be dropped.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Displays information on the object type.</td>
<td>table, views</td>
</tr>
<tr>
<td>Name</td>
<td>Displays the name of the object.</td>
<td>public.bs_operation_201804</td>
</tr>
<tr>
<td>Query</td>
<td>Displays the query that will be executed to drop the object.</td>
<td>DROP TABLE IF EXISTS public.a123</td>
</tr>
<tr>
<td>Status</td>
<td>Displays the status of the drop operation.</td>
<td>● To start  ● In progress  ● Completed  ● Error</td>
</tr>
<tr>
<td></td>
<td>● <strong>-</strong> To start: The drop operation yet to be initiated.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● <strong>-</strong> In progress: The object is currently being dropped.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● ✓ - Completed: The drop operation has been completed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● ✗ - Error: The object has not been dropped due to an error.</td>
<td></td>
</tr>
</tbody>
</table>
### Step 3
Select the required drop option:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cascade</td>
<td>Cascade drop operation drops their dependent objects and attributes. The dependent objects that are dropped will be removed from the Object Browser only after refresh operation is performed.</td>
</tr>
<tr>
<td>Atomic</td>
<td>Atomic drop operation drops all objects in case of success or drops none in case of a failure.</td>
</tr>
<tr>
<td>No selection</td>
<td>Un-selection of Atomic or Cascade does not drop dependent objects.</td>
</tr>
</tbody>
</table>

### Step 4
Click **Start**.

**Runs** - Displays the number of objects that are dropped from the total list of objects.

**Errors** - Displays the number of object that was not dropped due to errors.

### Step 5
Click **Stop** or close the **Drop Objects** dialog to stop the drop operation.

Refer to **Execute SQL Queries** section for information on copy, advanced copy, show/hide search bar, sort, and column reorder options.

---

**NOTE**

- Select part of cell content and press **Ctrl+C** or click ![Copy](atlas://IIcon/Check) to copy selected text from a cell.
- When you select multiple objects in object browser to drop, a batch drop window is opened and its menu icons are enabled in the menu bar. If you disconnect the database, the icons will remain disabled and will not be enabled even after reconnecting. You need to reselect the objects to drop and the selected objects will be available in the new terminal window.

### 4.22.3 Granting/Revoking Privileges

The batch grant/revoke operation allows you to select multiple objects to grant/revoke privileges. You can also perform batch grant/revoke operation on searched objects.

This feature is only available for OLAP, not for OLTP.

**NOTE**

Batch grant/revoke is allowed only with the same object type within that schema.
Follow the steps to grant/revoke privileges in a batch:

**Step 1** Press `Ctrl+left-click` (select objects one by one) or `Shift+left-click` (select objects in a bunch) to select the objects to grant/revoke privileges.

**Step 2** Right-click and select **Grant/Revoke**.

**Grant/Revoke** dialog box is displayed.

**Step 3** Refer to **Grant/Revoke Privilege** section to grant/revoke privilege.

----End

### 4.23 Personalizing Data Studio

#### 4.23.1 Overview

This section provides details on how to personalize Data Studio using preferences settings.

#### 4.23.2 General

This section provides details on how to personalize shortcut keys.

**Setting the Shortcut Keys**

You can customize the Data Studio shortcut keys as required.

Follow the steps to set or modify the shortcut keys:

**Step 1** Choose **Settings > Preferences** from the main menu.

The **Preferences** dialog box is displayed.

**Step 2** Choose **General > Shortcut Mapper**.

The **Shortcut Mapper** pane is displayed.

**Step 3** Select the required shortcut key and click **Modify**.

**Step 4** Enter the required shortcut key in the **Binding** text box.

For example, to change the shortcut key for **Step Into** from **F7** to **F6**, enter **F6** in the **Binding** text box.

**Step 5** Click **OK**.

The **Restart Data Studio** dialog box is displayed.

**NOTE**

- Multiple shortcut keys can be modified before restarting Data Studio.

**Step 6** Click **Yes** to restart Data Studio. If any export, import or execution operations are in progress, then the **Restart Confirmation** dialog box is displayed.
Step 7 Click OK to close running jobs and restart or click Cancel to abort restart operation.

----End

Follow the steps to remove the shortcut keys:

Step 1 Choose Settings > Preferences from the main menu. The Preferences dialog box is displayed.

Step 2 Choose General > Shortcut Mapper. The Shortcut Mapper pane is displayed.

Step 3 Select the required shortcut key and click Unbind Key.

Step 4 Click Ok.

The Restart Data Studio window is displayed.

**NOTE**

Multiple shortcut keys can be removed before restarting Data Studio

Step 5 Click Yes to restart Data Studio. If any export, import or execution operations are in progress, then the Process is running dialog box is displayed.

Step 6 Click OK to wait for operations to complete or click Force Restart to discard operations.

----End

Follow the steps to restore the default shortcut keys:

Step 1 Choose Settings > Preferences from the main menu. The Preferences dialog box is displayed.

Step 2 Choose General > Shortcut Mapper. The Shortcut Mapper pane is displayed.

Step 3 Click Restore Defaults. For more information on default shortcut keys, refer to Data Studio Right-Click Menus.

Step 4 Click Ok.

The Restart Data Studio window is displayed.

Step 5 Click Yes to restart Data Studio. If any export, import or execution operations are in progress, the Process is running dialog box displays.

Step 6 Click OK to wait for operations to complete or click Force Restart to discard operations.

----End

**Shortcut Keys**

Data Studio supports keyboard short cut keys similar to other windows based application. The following table lists some of the shortcut keys for effective usage
of the functionalities provided by Data Studio. To customize the shortcut keys, refer to **Setting the Shortcut Keys**.

<table>
<thead>
<tr>
<th>Function</th>
<th>Shortcut Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sorts the result sets of visual charts, edit tables, and queries in ascending, descending, or server receiving order</td>
<td>Alt+Click</td>
</tr>
<tr>
<td>Help menu</td>
<td>Alt+H</td>
</tr>
<tr>
<td>Save the SQL script</td>
<td>Ctrl+S</td>
</tr>
<tr>
<td>Edit menu</td>
<td>Alt+E</td>
</tr>
<tr>
<td>Compiling/Executing SQL Terminal Statements</td>
<td>Ctrl+Enter</td>
</tr>
<tr>
<td>Search and Replace</td>
<td>Ctrl+F</td>
</tr>
<tr>
<td>Search for the previous one</td>
<td>Shift+F3</td>
</tr>
<tr>
<td>Search for the next one</td>
<td>F3</td>
</tr>
<tr>
<td>Redoing</td>
<td>Ctrl+Y</td>
</tr>
<tr>
<td>On the Edit Table Data tab page, copy Execution Time and Status</td>
<td>Ctrl+Shift+K</td>
</tr>
<tr>
<td>Copy the database object from the automatic recommendation list</td>
<td>Alt+U</td>
</tr>
<tr>
<td>Open the Call Stack, the Breakpoints pane, and the Variables pane</td>
<td>Alt+V</td>
</tr>
<tr>
<td>Open the SQL script</td>
<td>Ctrl+O</td>
</tr>
<tr>
<td>Step Skip</td>
<td>F8</td>
</tr>
<tr>
<td>Step into</td>
<td>F7</td>
</tr>
<tr>
<td>Single step exit</td>
<td>Shift+F7</td>
</tr>
<tr>
<td>Comment out or cancel the comment line</td>
<td>Ctrl+/</td>
</tr>
<tr>
<td>Locate the first element in the Object Browser</td>
<td>Alt+Page Up or Alt+Home</td>
</tr>
<tr>
<td>Locate the last element in the Object Browser</td>
<td>Alt+Page Down or Alt+End</td>
</tr>
<tr>
<td>Locate to row</td>
<td>Ctrl+G</td>
</tr>
<tr>
<td>Disconnect the connection</td>
<td>Ctrl+Shift+D</td>
</tr>
<tr>
<td>Formatting (SQL and PL/SQL)</td>
<td>Ctrl+Shift+F</td>
</tr>
<tr>
<td>Change the value to uppercase</td>
<td>Ctrl+Shift+U</td>
</tr>
<tr>
<td>Change the value to lowercase</td>
<td>Ctrl+Shift+L</td>
</tr>
<tr>
<td>Function</td>
<td>Shortcut Key</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Updates the cells or columns in the Edit Table Data, Properties, and Results windows. Click the cell or column header to enable this option.</td>
<td>F2</td>
</tr>
<tr>
<td>Close the PL/SQL Viewer tab page, Table Data View tab page, Execute Query tab page, or Properties tab page.</td>
<td>Shift+F4</td>
</tr>
<tr>
<td>Continue the PL/SQL debugging</td>
<td>F9</td>
</tr>
<tr>
<td>Shearing</td>
<td>Ctrl+X</td>
</tr>
<tr>
<td>Copy Object Browser or the name of the object modified in the terminal. Copy the selected data from the Terminal, Result, Table Data, or Edit Table Data tab page.</td>
<td>Ctrl+C</td>
</tr>
<tr>
<td>Copy the data on the Result, Table Data, or Edit Table Data tab page. The data contains/does not contain the column title and row number.</td>
<td>Ctrl+Shift+C</td>
</tr>
<tr>
<td>Copy the query result on the Edit Table Data tab page</td>
<td>Ctrl+Alt+C</td>
</tr>
<tr>
<td>Copy the content on the Variable tab page</td>
<td>Alt+K</td>
</tr>
<tr>
<td>Copy the content on the Call Stack tab page</td>
<td>Alt+J</td>
</tr>
<tr>
<td>Copy the content on the Breakpoint tab page</td>
<td>Alt+Y</td>
</tr>
<tr>
<td>Visualized interpretation plan</td>
<td>Alt+Ctrl+X</td>
</tr>
<tr>
<td>Online help (displaying the user manual)</td>
<td>F1</td>
</tr>
<tr>
<td>Template</td>
<td>Alt+Ctrl+Space</td>
</tr>
<tr>
<td>Switch to the first SQL Terminal tab page</td>
<td>Alt+S</td>
</tr>
<tr>
<td>Select All</td>
<td>Ctrl+A</td>
</tr>
<tr>
<td>Setting menu</td>
<td>Alt+G</td>
</tr>
<tr>
<td>Refresh (in the Object Browser area)</td>
<td>F5</td>
</tr>
<tr>
<td>Search Object</td>
<td>Ctrl+Shift+S</td>
</tr>
<tr>
<td>Debugging menu</td>
<td>Alt+D</td>
</tr>
<tr>
<td>Debugging template</td>
<td>F10</td>
</tr>
<tr>
<td>Debugging the Database Object</td>
<td>Ctrl+D</td>
</tr>
<tr>
<td>Highlight Object Browser</td>
<td>Alt+X</td>
</tr>
<tr>
<td>File menu</td>
<td>Alt+F</td>
</tr>
<tr>
<td>Function</td>
<td>Shortcut Key</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>Creating a connection</td>
<td>Ctrl+N</td>
</tr>
<tr>
<td>Running menu</td>
<td>Alt+R</td>
</tr>
<tr>
<td>Switch between the SQL Terminal tab page</td>
<td>Ctrl+Page Up or Ctrl+Page Down</td>
</tr>
<tr>
<td>Expand/Collapse All Objects</td>
<td>Ctrl+M</td>
</tr>
<tr>
<td>Pastes</td>
<td>Ctrl+V</td>
</tr>
<tr>
<td>Collapsible object browsing navigation tree</td>
<td>Alt+Q</td>
</tr>
<tr>
<td>Execute</td>
<td>Ctrl+E</td>
</tr>
<tr>
<td>Execution plan and expense</td>
<td>Ctrl+Shift+X</td>
</tr>
<tr>
<td>Stop the query in the running state</td>
<td>Shift+Esc</td>
</tr>
<tr>
<td>Comment/Cancel the comment line or the entire segment</td>
<td>Ctrl+Shift+/</td>
</tr>
<tr>
<td>List of automatically recommended database objects</td>
<td>Ctrl+Space</td>
</tr>
</tbody>
</table>

### 4.23.3 Editor

This section provides details on how to personalize syntax coloring, SQL history information, templates, and formatter.

#### Syntax Coloring

Follow the steps to customize the SQL highlight color:

**Step 1** Choose **Settings > Preferences** from the main menu.

The **Preferences** dialog box is displayed.

**Step 2** Choose **Editor > Syntax Coloring**.

The **Syntax Coloring** pane is displayed.

**Step 3** Click the color button to customize the color for the type of syntax.

For example, click ![Blue](blue.png) to customize the color for **Strings**. The color picker dialog box is displayed.

Use the color picker to set the required color for a specific syntax category. You can choose basic colors or define custom colors in the color picker.

*NOTE*

Click **Restore Defaults** from **Syntax Coloring** pane to reset to default color scheme.

**Step 4** Click **OK**. The **Restart Data Studio** dialog box is displayed.
Step 5 Click Yes to restart Data Studio. If any export, import or execution operations are in progress, then Data Studio displays the Process is running dialog box.

Step 6 Click Force Restart to discard operations and restart Data Studio. Click OK to continue performing operations.

NOTE

The Preferences.prefs file contains the custom color settings. If the file is corrupted, Data Studio will display the default values.

The custom color(s) will be set after you restart Data Studio.

---End

SQL History

You can customize Data Studio to set the number of SQL history count that can be made available and also the number of characters for the query for each of the query saved in SQL history.

Follow the steps to customize the number of executed queries and number of characters in the query to be saved in SQL History:

Step 1 Choose Settings > Preferences from the main menu.

The Preferences dialog box is displayed.

Step 2 Choose Editor > SQL History.

The SQL History pane is displayed.

Step 3 Set the number of queries to be saved in SQL History Count field.

NOTE

Minimum value is 1 and maximum is 1000. The current value set for this preference is displayed.

Step 4 Set the number of characters to be allowed in each query that is saved in the SQL History in the SQL Query Characters field.

NOTE

Minimum value is 1 and maximum is 1000. Enter "0" in this field to set no character limit. The current value set for this preference will be displayed.

Step 5 Click Apply.

Step 6 Click OK.
NOTE

- Click Restore Defaults from SQL History pane to reset to default value.
- The default value for SQL History Count is 50 and SQL Query Characters it is 1000.
- If the new value entered is lesser than the old value and if there is going to be a data loss, then a message is displayed informing about the data loss and if you would like to continue with the operation.
- If there are unsaved changes and you navigate away from this pane, then a message displays to state that there are unsaved changes.
- Pinned queries are not affected by the changes made to the SQL History Count field. Example: If the number of pinned queries is 50 and the SQL History Count is set to 25, then SQL History will show 50 pinned queries.
- The SQL Query Characters changes affects only queries added post the configuration change.

Adding Templates

You can customize Data Studio to create new, edit existing, and remove templates. Refer to the Using Templates section for detailed information on templates.

NOTE

Restoring the settings to default removes all user defined templates from the list.

Follow the steps to create templates:

Step 1  Choose Settings > Preferences from the main menu.
  The Preferences dialog box is displayed.

Step 2  Choose Editor > Templates.
  The Templates pane is displayed.

Step 3  Click New.

Step 4  Enter a name for the template in the Name field.

Step 5  Enter description in the Description field.

Step 6  Enter the SQL statement pattern in the Pattern field.

NOTE

The text entered in Pattern field will be syntax highlighted.

Step 7  Click OK.

Modifying Templates

Follow the steps to edit templates:

Step 1  Choose Settings > Preferences from the main menu.
  The Preferences dialog box is displayed.
Step 2  Choose Editor > Templates.

The Templates pane is displayed.

Step 3  Click Edit.

Step 4  Edit the name in the Name field, if required.

Step 5  Edit the description in the Description field, if required.

Step 6  Edit the SQL statement pattern in the Pattern field, if required.

💡 NOTE

The text entered in Pattern field will be syntax highlighted.

Step 7  Click OK.

----End

Removing Templates

Follow the steps to remove templates:

Step 1  Choose Settings > Preferences from the main menu.

The Preferences dialog box is displayed.

Step 2  Choose Editor > Templates.

The Templates pane is displayed.

Step 3  Select the template to be removed, and click Remove.

Removes the template from the Templates list.

💡 NOTE

Default templates that are removed can be added back using Restore Removed option. It will restore the template to the last updated change. Restore Removed option is not applicable to user defined templates.

----End

Reverting to Default Templates

Follow the steps to revert to default templates:

Step 1  Choose Settings > Preferences from the main menu.

The Preferences dialog box is displayed.

Step 2  Choose Editor > Templates.

The Templates pane is displayed.

Step 3  Select at least one default template that is modified to revert to default template settings.

Step 4  Click Revert to Default.

----End
Formatter

You can customize Data Studio to set the tab width and convert tab to spaces while performing indent and un-indent operation. Refer to **Indent/Unindent Lines** section to perform indent/un-indent operation and replace tab with spaces.

Follow the steps to customize the indent size and convert tab to spaces:

**Step 1**  Choose **Settings > Preferences** from the main menu.

The **Preferences** dialog box is displayed.

**Step 2**  Choose **Editor > Formatter**.

The **Formatter** pane is displayed.

**Step 3**  Select the **Insert Space** option to replace tab with spaces or **Insert Tab** to add/remove tabs while indenting/un-indenting lines.

**Step 4**  Enter the indent size in **Indent Size**. Based on the number specified in this field, the indent/un-indent/space length is defined.

----End

Transaction

Follow the steps to edit Transaction settings:

**Step 1**  Choose **Settings > Preferences** from the main menu.

The **Preferences** dialog box is displayed.

**Step 2**  Choose **Editor > Transaction**.

The **Transaction** pane is displayed.

**Step 3**  In **Auto Commit** window select **Enable** to switch on the auto commit feature. In this case, commit and rollback button will be disabled.

- Transaction will be committed automatically.

- Select **Disable** to switch off the auto commit feature. Commit and Rollback button can be used manually for committing or reverting changes.
4.23.4 Environment

Session Setting

Follow the steps to set Data Studio and file encoding:

Step 1 Choose Settings > Preferences from the main menu.

Step 2 Choose Environment > Session Setting.

Step 3 Select the required Data Studio encoding from Data Studio Encoding drop-down.

Step 4 Select the file encoding from File Encoding field.

NOTE

Data Studio supports only UTF-8 and GBK file encoding types.

Step 5 Click OK. The Restart Data Studio dialog box is displayed.

Step 6 Click Yes to restart Data Studio. If any export, import or execution operations are in progress, then Data Studio displays the Process is running dialog box.

Step 7 Click Force Restart to discard operations and restart Data Studio. Click OK to continue performing operations.

NOTE

- Click Restore Defaults from Session Setting pane to reset to default values. The default value for Data Studio Encoding and File Encoding is UTF-8.

SQL Assistant

Follow the steps to enable/disable SQL Assistant tool:

Step 1 Choose Settings > Preferences from the main menu.

Step 2 Choose Environment > Session Setting.

Step 3 Select Enable/Disable from SQL Assistant section.

Step 4 Click OK.
NOTE
Click Restore Defaults from Session Setting pane to reset to default value. The default value for SQL Assistant is Enable.

---End

Query/Function/Procedure Backup

Refer to the Backup Unsaved Queries/Functions/Procedures section for information on backup feature provided by Data Studio.

Follow the steps to enable/disable backup of unsaved data in SQL Terminal/PL/SQL Viewer:

Step 1 Choose Settings > Preferences from the main menu.

The Preferences dialog box is displayed.

Step 2 Choose Environment > Session Setting.

The Session Setting pane is displayed.

Step 3 Select/un-select Auto Save from Auto Save section.

Step 4 Set the time interval to backup the data in Interval field.

Step 5 Click OK.

NOTE
Click Restore Defaults from Session Setting pane to reset to default value. Backup of data will be enabled by default with 5 minutes as the default time interval.

---End

Follow the steps to enable/disable data encryption of saved data:

Step 1 Choose Settings > Preferences from the main menu.

The Preferences dialog box is displayed.

Step 2 Choose Environment > Session Setting.

The Session Setting pane is displayed.

Step 3 Select/un-select Encryption from Auto Save section.

Step 4 Click OK.

NOTE
Click Restore Defaults from Session Setting pane to reset to default value. Encryption will be enabled by default.

---End

Follow the steps to set the size of Import Table Data Limit/Import File Data Limit.

Step 1 Choose Settings > Preferences from the main menu.

The Preferences dialog box is displayed.
Step 2  Choose Environment > Session Setting.

The Session Setting pane is displayed.

In File Limit section Import Table Data Limit and Import File Data Limit parameters are displayed.

Import Table Data Limit value defines the maximum size of the table data to be imported.

Import File Data Limit value defines the maximum size of the file to be imported.

Step 3  Click OK.

NOTE

Mentioned values in the earlier mentioned screenshot are the default values.

----End

4.23.5 Export

This section provides details on how to personalize export DDL operation.

Export DDL

You can set preference to include tablespace in DDL while exporting DDL using the Export DDL setting.

Set include tablespace in DDL:
Step 1 Choose Settings > Preferences from the main menu.

The Preferences dialog box is displayed.

Step 2 Choose Export/Import > Export DDL.

The Export DDL pane is displayed.

Step 3 Select Include Tablespace in DDL to include the tablespace while exporting DDL.

Step 4 Click OK.

- Click Restore Defaults from Export DDL pane to reset to default values. The default value is Include Tablespace in DDL.

----End

4.23.6 Result Management

This section provides details on how to personalize the column width, number of records to be fetched in the query results, and result copy of column header or row number using the Query Results setting. It also provides details on how to display the data encoding in edit, view and query results table.

Query Results

Set column width of query results:

Step 1 Choose Settings > Preferences from the main menu.

The Preferences dialog box is displayed.

Step 2 Choose Result Management > Query Results.

The Query Results pane is displayed.

Step 3 Select the required option.

Column Width customization options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content Length</td>
<td>Selecting this option enables you to set the column width based on the content length of the column.</td>
</tr>
<tr>
<td>Custom Length</td>
<td>Selecting this option enables you to set the column width based on the value entered in this field. NOTE This column accepts value between 100 and 500.</td>
</tr>
</tbody>
</table>

Step 4 Click OK.

- Click Restore Defaults from Query Results pane to reset to default values. The default value is Content Length.

----End
Set the number of records to be fetched in the query results:

**Step 1** Choose **Settings > Preferences** from the main menu.

The **Preferences** dialog box is displayed.

**Step 2** Choose **Result Management > Query Results**.

The **Query Results** pane is displayed.

**Step 3** Select the required option.

<table>
<thead>
<tr>
<th>Option</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fetch All records</td>
<td>Selecting this option enables you to fetch all the records in the query results.</td>
</tr>
</tbody>
</table>
| Fetch custom number of records | Selecting this option enables you to set the number of records that needs to be fetched in the query results.  
**NOTE** This column accepts value between 100 and 5000. |

**Step 4** Click **OK**.

**NOTE**
Click **Restore Defaults** from **Query Results** pane to reset to default values. The default value is **Fetch custom number of records (1000)**.

----End

Set preference to copy column name and row number from query results:

**Step 1** Choose **Settings > Preferences** from the main menu.

The **Preferences** dialog box is displayed.

**Step 2** Choose **Result Management > Query Results**.

The **Query Results** pane is displayed.

**Step 3** Select the required option.

<table>
<thead>
<tr>
<th>Option</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Include column header</td>
<td>Selecting this option enables you to copy column headers from the query results.</td>
</tr>
<tr>
<td>Include row number</td>
<td>Selecting this option enables you to copy row numbers from the query results.</td>
</tr>
</tbody>
</table>

**Step 4** Click **OK**.
NOTE

Click **Restore Defaults** from **Query Results** pane to reset to default values. The default value is **Include column header**.

---End

Edit Table Data

Set save behavior of edit table data operation:

**Step 1** Choose **Settings > Preferences** from the main menu.

The **Preferences** dialog box is displayed.

**Step 2** Choose **Result Management > Edit Table Data**.

The **Edit Table Data** pane is displayed. Select the required option:

**Table 4-24 Edit table data**

<table>
<thead>
<tr>
<th>Server Type</th>
<th>Auto Commit</th>
<th>Reuse Connection</th>
<th>Table Data Save Option</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>DWS</td>
<td>ON</td>
<td>ON</td>
<td>Save Valid Data</td>
<td>All the valid data will be saved and committed. Incorrect data will be omitted.</td>
</tr>
<tr>
<td>DWS</td>
<td>ON</td>
<td>ON</td>
<td>Do Not Save</td>
<td>If an error occurs, no data will be saved.</td>
</tr>
<tr>
<td>DWS</td>
<td>ON</td>
<td>OFF</td>
<td>Save Valid Data</td>
<td>All the valid data will be saved and committed. Incorrect data will be omitted.</td>
</tr>
<tr>
<td>DWS</td>
<td>ON</td>
<td>OFF</td>
<td>Do Not Save</td>
<td>If an error occurs, no data will be saved.</td>
</tr>
<tr>
<td>Server Type</td>
<td>Auto Commit</td>
<td>Reuse Connection</td>
<td>Table Data Save Option</td>
<td>Behavior</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>------------------</td>
<td>-------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>DWS</td>
<td>OFF</td>
<td>ON</td>
<td>Save Valid Data</td>
<td>If an error occurs, no data will be saved. Perform Commit/Rollback to proceed further.</td>
</tr>
<tr>
<td>DWS</td>
<td>OFF</td>
<td>ON</td>
<td>Do Not Save</td>
<td>If an error occurs, no data will be saved. Perform Commit/Rollback to proceed further.</td>
</tr>
<tr>
<td>Cluster</td>
<td>ON</td>
<td>ON</td>
<td>Save Valid Data</td>
<td>All the valid data will be saved and committed. Incorrect data will be omitted.</td>
</tr>
<tr>
<td>Cluster</td>
<td>ON</td>
<td>ON</td>
<td>Do Not Save</td>
<td>If an error occurs, no data will be saved.</td>
</tr>
<tr>
<td>Cluster</td>
<td>ON</td>
<td>OFF</td>
<td>Save Valid Data</td>
<td>All the valid data will be saved and committed. Incorrect data will be omitted.</td>
</tr>
<tr>
<td>Cluster</td>
<td>ON</td>
<td>OFF</td>
<td>Do Not Save</td>
<td>If an error occurs, no data will be saved.</td>
</tr>
<tr>
<td>Server Type</td>
<td>Auto Commit</td>
<td>Reuse Connection</td>
<td>Table Data Save Option</td>
<td>Behavior</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>------------------</td>
<td>------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>Cluster</td>
<td>OFF</td>
<td>ON</td>
<td>Save Valid Data</td>
<td>All the valid data will be saved. If any error occurs it roll backs the entire transaction and it saves valid data further.</td>
</tr>
<tr>
<td>Cluster</td>
<td>OFF</td>
<td>ON</td>
<td>Do Not Save</td>
<td>If an error occurs, no data will be saved.</td>
</tr>
<tr>
<td>Standalone</td>
<td>ON</td>
<td>ON</td>
<td>Save Valid Data</td>
<td>All the valid data will be saved and committed. Incorrect data will be omitted.</td>
</tr>
<tr>
<td>Standalone</td>
<td>ON</td>
<td>ON</td>
<td>Do Not Save</td>
<td>If an error occurs, no data will be saved.</td>
</tr>
<tr>
<td>Standalone</td>
<td>ON</td>
<td>OFF</td>
<td>Save Valid Data</td>
<td>All the valid data will be saved and committed. Incorrect data will be omitted.</td>
</tr>
<tr>
<td>Standalone</td>
<td>ON</td>
<td>OFF</td>
<td>Do Not Save</td>
<td>If an error occurs, no data will be saved.</td>
</tr>
<tr>
<td>Standalone</td>
<td>OFF</td>
<td>ON</td>
<td>Save Valid Data</td>
<td>All the valid data will be saved. Incorrect data will be omitted.</td>
</tr>
</tbody>
</table>
### Result Data Encoding

You can enable/disable to display the data encoding type in edit, view, and query results window.

Follow the steps to modify display of encoding option:

**Step 1** Choose **Settings > Preferences** from the main menu.

The **Preferences** dialog box is displayed.

**Step 2** Choose **Result Management > Query Results**.

The **Query Results** pane is displayed.

**Step 3** Select **Include result data encoding** to include the **Encoding** drop-down in edit, view, and query results table.

**Step 4** Click **OK**.

**NOTE**
- Click **Restore Defaults** from **Result Management** pane to reset to default values. **Include result data encoding** will be unselected by default.
- Edit table, view table properties and query execution must be performed again to apply the changes.

---

### 4.23.7 Security

This section provides details on how to personalize password and security disclaimer display.

**Save Password Permanently**

You can enable/disable to display the permanent option to save password in the connection window.

Follow the steps to modify display of permanent save password option:
Step 1 Choose Settings > Preferences from the main menu.

The Preferences dialog box is displayed.

Step 2 Choose Security > Password.

The Password pane is displayed.

Step 3 Select the required option. Refer table below to understand the customization options available:

<table>
<thead>
<tr>
<th>Option</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Selecting this option enables you to view the &quot;Permanently&quot; save password option from the Save Password drop-down in the connection window.</td>
</tr>
<tr>
<td>No</td>
<td>Selecting this option removes the &quot;Permanently&quot; save password option from the Save Password drop-down in the connection window. Selecting this option removes the saved passwords.</td>
</tr>
</tbody>
</table>

Step 4 Click OK. The Restart Data Studio dialog box is displayed.

Step 5 Click Yes to restart Data Studio. If any export, import or execution operations are in progress, then Data Studio displays the Process is running dialog box.

Step 6 Click Force Restart to discard operations and restart Data Studio. Click OK to continue performing operations.

NOTE

Click Restore Defaults from Password pane to reset to default values. The default value is No.

----End

Password Expiry

This section provides details on how to continue/discontinue working with Data Studio once password expires using the Password setting.

Follow the steps to modify the behavior of Data Studio once password expires:

Step 1 Choose Settings > Preferences from the main menu.

The Preferences dialog box is displayed.

Step 2 Choose Security > Password.

The Password pane is displayed.

Step 3 Select the required option. Refer table below to understand the customization options available:
<table>
<thead>
<tr>
<th>Option</th>
<th>Outcome</th>
</tr>
</thead>
</table>
| Yes    | Selecting this option allows you to login to Data Studio after the password has expired.  
**NOTE**  
A message displays informing you that the password has expired and some operations may not work as expected in the following scenarios:  
- Establishing a new connection.  
- Editing a connection.  
- Connecting to a database while creating the database when no other database is connected in that connection profile.  
- Connecting to a database when no other database is connected in that connection profile. |
| No     | Selecting this option will not allow you to login to Data Studio once the password has expired. A message displays informing you that the password has expired. |

**Step 4** Click **OK**. The **Restart Data Studio** dialog box is displayed.

**Step 5** Click **Yes** to restart Data Studio. If any export, import or execution operations are in progress, then Data Studio displays the **Process is running** dialog box.

**Step 6** Click **Force Restart** to discard operations and restart Data Studio. Click **OK** to continue performing operations.

**NOTE**

The default value is **Yes**.

---End

**Security Disclaimer**

You can enable/disable to display the security disclaimer for any insecure connection/file operations.

Follow the steps to modify the display of security disclaimer:

**Step 1** Choose **Settings > Preferences** from the main menu.

The **Preferences** dialog box is displayed.

**Step 2** Choose **Security > Security Disclaimer**.

The **Security Disclaimer** pane is displayed.

**Step 3** Select the required option. Refer table below to understand the customization options available:

<table>
<thead>
<tr>
<th>Option</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>Selecting this option displays the security disclaimer each time you try to establish an insecure connection or perform a file operation.</td>
</tr>
</tbody>
</table>
### 4.24 Performance Specification

Data Studio's performance to load and perform an operation on the object browser directly depends on the number of objects to be loaded. These include tables, views, columns and so on.

The memory consumption also depends on the number of objects loaded.

To improve the performance of loading objects and of memory usage efficiency, it is recommended to split the objects across multiple namespaces and avoid having skewed namespaces with a very large number of objects. By default, Data Studio loads the namespaces in the `search_path` set for the user logged in. Other namespaces and the contained objects are loaded only when needed.

To improve performance load all objects rather than loading them based on user privilege. **Minimum Privileges Requirement** table provides information on the minimum access required for objects to be listed in Object Browser.

#### Table 4-25 Minimum Privileges Requirement

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Privilege Types</th>
<th>Object Browser - Minimum Privilege Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Create, Connect, Temporary/Temp, All</td>
<td>Connect</td>
</tr>
<tr>
<td>Schema</td>
<td>Create, Usage, All</td>
<td>Usage</td>
</tr>
<tr>
<td>Table</td>
<td>Select, Insert, Update, Delete, Truncate, References, All</td>
<td>Select</td>
</tr>
<tr>
<td>Object Type</td>
<td>Privilege Types</td>
<td>Object Browser - Minimum Privilege Type</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>Column</td>
<td>Select, Insert, Update, References, All</td>
<td>Select</td>
</tr>
<tr>
<td>View</td>
<td>Select, Insert, Update, Delete, Truncate, References, All</td>
<td>Select</td>
</tr>
<tr>
<td>Sequences</td>
<td>Usage, Select, Update, All</td>
<td>Usage</td>
</tr>
<tr>
<td>Function</td>
<td>Execute, All</td>
<td>Execute</td>
</tr>
<tr>
<td>Tablespace</td>
<td>Create, All</td>
<td>Create</td>
</tr>
</tbody>
</table>

To improve performance of **Find/Find and Replace** operation, it is recommended that split lines that have more than 10000 characters in a single line into multiple smaller lines.

The following observations and outcomes of the performance tests done will help understand the performance aspects of Data Studio more clearly:

| Recommended maximum memory that can be configured (Current release version) | 1.4 GB |
| Performance with 150K tables and 150K views with 3 columns each at maximum memory configuration: | |
| > Time taken to refresh the namespace in the object browser | 15s |
| > Time taken for initial loading and expanding of all table/views in the object browser | 90s-120s |
| > Time taken for subsequent loading and expanding of all table/views in the object browser | <10s |
| > Total Memory occupied | 700 MB |

**NOTE**

This representative performance data is included for reference only. Actual performance may vary depending on the usage scenarios.

### 4.25 Security Management
4.25.1 Overview

**NOTICE**

Ensure that the operating system and the required software's (refer to System Requirements for more details) are updated with the latest patches to prevent vulnerabilities and other security issues.

This section provides the security management information for Data Studio.

4.25.2 Login History

The following information is critical to manage security for Data Studio:

When you log into the database, Data Studio displays a pop-up with details of the last successful login and failure attempts between the last two successful logins for you on the logged database.

![RECENT LOGIN ACTIVITY](image)

**NOTE**

- If the pop-up displays the message "Last login details not available", then it implies that the connected database does not support the last login display feature.

4.25.3 Password Expiry Notification

The following information is critical to manage security for Data Studio:

- Your password will expire within 7 days from the date of notification. If the password expires, contact the database administrator to reset the password.
- The password must be changed every 90 days.

4.25.4 Securing the Application In-Memory Data

The following information is critical to manage security for Data Studio:

While running Data Studio in trusted environment, user must ensure to prevent malicious software to scan or access the memory which is used to store application data including sensitive information.
Alternatively, you can choose Do Not Save while connecting to the database, so that password does not get saved in the memory.

4.25.5 Data Encryption for Saved Data

You can ensure encryption of auto saved data by enabling encryption option from Preferences page.

Refer to Query/Function/Procedure Backup section for steps to encrypt the saved data.

4.25.6 SQL History

The following information is critical to manage security for Data Studio:

- SQL History scripts are not encrypted.
- The SQL History list does not display sensitive queries that contain the following keywords:
  - Alter Role
  - Alter User
  - Create Role
  - Create User
  - Identified by
  - Password
- Few query syntax examples are listed below:
  - ALTER USER name [ [ WITH ] option [ ... ] ]
  - CREATE USER name [ [ WITH ] option [ ... ] ]
  - CREATE ROLE name [ [ WITH ] option [ ... ] ]
  - ALTER ROLE name [ [ WITH ] option [ ... ] ]

4.25.7 SSL Certificates

The information on using SSL certificates is included only for reference purposes. For details on the certificates and for security guidelines for managing the certificates and related files, refer the database server documentation.

Data Studio can connect to the database using the Secure Sockets Layer [SSL] option. The following files are required to add a connection.

<table>
<thead>
<tr>
<th>#</th>
<th>Certificate/Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Client SSL Certificate</td>
<td>Provided by System/Database Administrator</td>
</tr>
<tr>
<td>2</td>
<td>Client SSL Key</td>
<td>Provided by System/Database Administrator</td>
</tr>
<tr>
<td>3</td>
<td>Root Certificate</td>
<td>Provided by System/Database Administrator</td>
</tr>
</tbody>
</table>
SSL Certificate Generation and Server Configuration

Follow the steps to generate the certificate:

**Step 1 Establish a CA environment** - Assume that user `omm` has been created and the CA path is `test`.

Log in to SUSE Linux as user `root` and switch to user `omm`.

Execute the following command:

```bash
mkdir test
cd /etc/ssl
cp openssl.cnf ~/test
cd ~/test
```

Copy the configuration file `openssl.cnf` to `test`.

Command:

```bash
cp openssl.cnf ~/test
cd ~/test
```

Establish the CA environment under the `test` folder.

Create folder in `demoCA./demoCA/newcerts./demoCA/private` path.

Command:

```bash
mkdir ./demoCA ./demoCA/newcerts ./demoCA/private
chmod 777 ./demoCA/private
```

Create the `serial` file and write it to `01`.

Command:

```bash
echo '01'>./demoCA/serial
```

Create the `index.txt` file.

Command:

```bash
touch /home/omm/test/demoCA/index.txt
```

Modify parameters in the `openssl.cnf` configuration file.

Command:

```bash
dir = /home/omm/test/demoCA
default_md = sha256
```

The CA environment has been established.

**Step 2 Generate a root private key** - Generate a CA private key.

Command:

```bash
openssl genrsa -aes256 -out demoCA/private/cakey.pem 2048
```

Generating RSA private key, 2048 bit long modulus.

**Step 3 Generate a root certificate request file** - CA root certificate application file named `server.req`.

Command:

```bash
openssl req -config openssl.cnf -new -key demoCA/private/cakey.pem -out demoCA/careq.pem
```

Enter pass phrase for `demoCA/private/cakey.pem`
Enter the root private key password **Gauss@MppDB**.

You are about to be asked to enter information that will be incorporated into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank.

For some fields there will be a default value, enter '.' to leave the field blank. Enter the following information in the generated server certificate and client certificate.

- **Country Name (2 letter code)** [AU]: CN
- **State or Province Name (full name)** [Some-State]: shanxi
- **Locality Name (eg, city)** []: xian
- **Organization Name (eg, company)** [Internet Widgits Pty Ltd]: Abc
- **Organizational Unit Name (eg, section)** []: hello
- **Common name can be any name**
- **Common Name (eg, YOUR name)** []: world
- **Email is optional.**
- **Email Address** []: 
- **A challenge password** []: 
- **An optional company name** []:

**Step 4 Generate a self-signed root certificate.**

Command:

```plaintext
openssl ca -config openssl.cnf -out demoCA/cacert.pem -keyfile demoCA/private/cakey.pem -selfsign -infiles demoCA/careq.pem
```

Using configuration from openssl.cnf

Enter pass phrase for **demoCA/private/cakey.pem**

Enter the root private key password **Gauss@MppDB**.

Check that the request matches the signature.

Signature ok
Certificate Details:
  Serial Number: 1 (0x1)
  Validity
    Not Before: Feb 28 02:17:11 2017 GMT
    Not After : Feb 28 02:17:11 2018 GMT
  Subject:
    countryName = CN
    stateOrProvinceName = shanxi
    organizationName = Abc
    organizationalUnitName = hello
    commonName = world
  X509v3 extensions:
    X509v3 Basic Constraints: CA:FALSE
    Netscape Comment: OpenSSL Generated Certificate
  X509v3 Subject Key Identifier:
  X509v3 Authority Key Identifier:
  Certificate is to be certified until Feb 28 02:17:11 2018 GMT (365 days)
  Sign the certificate? [y/n]: y
1 out of 1 certificate requests certified, commit? [y/n]: y
Write out database with 1 new entries
Data Base Updated

A CA root certificate named **demoCA/cacert.pem** has been issued.
Step 5  Generate a private key for the server certificate

Generate a private key file named server.key.

Command:

```
openssl genrsa -aes256 -out server.key 2048
```

Step 6  Generate a server certificate request file

Generate a server certificate request file server.req.

Command:

```
openssl req -config openssl.cnf -new -key server.key -out server.req
```

Enter pass phrase for server.key:

You are about to be asked to enter information that will be incorporated into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

Set the following information and make sure that it is same as that when CA is created.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Country Name (2 letter code) [AU]:</td>
<td>CN</td>
</tr>
<tr>
<td>State or Province Name (full name)</td>
<td>shanxi</td>
</tr>
<tr>
<td>Locality Name (eg, city) [xian]</td>
<td></td>
</tr>
<tr>
<td>Organization Name (eg, company)</td>
<td>Abc</td>
</tr>
<tr>
<td>Organizational Unit Name (eg, section) [hello]</td>
<td></td>
</tr>
<tr>
<td>Common Name (eg, YOUR name) [world]</td>
<td></td>
</tr>
<tr>
<td>Email Address []:</td>
<td></td>
</tr>
</tbody>
</table>

A challenge password []:

-- The following information is optional.
An optional company name []:

Step 7  Generate a server certificate - Change the demoCA/index.txt.attr attribute to no.

vi demoCA/index.txt.attr

Issue the generated server certificate request file. After it is issued, an official server certificate server.crt is generated.

```
openssl ca -config openssl.cnf -in server.req -out server.crt -days 3650 -md sha256
```

Using configuration from /etc/ssl/openssl.cnf

Enter pass phrase for ./demoCA/private/cakey.pem:

Check that the request matches the signature

```
Signature ok
Certificate Details:
Serial Number: 2 (0x2)
Validity
Not Before: Feb 27 10:11:12 2017 GMT
Not After : Feb 25 10:11:12 2027 GMT
Subject:
countryName = CN
stateOrProvinceName = shanxi
```
organizationName = Abc
organizationalUnitName = hello
commonName = world
X509v3 extensions:
X509v3 Basic Constraints:
CA:FALSE
Netscape Comment:
OpenSSL Generated Certificate
X509v3 Subject Key Identifier:
X509v3 Authority Key Identifier:
Certificate is to be certified until Feb 25 10:11:12 2027 GMT (3650 days)
-- Choose y to sign and issue the certificate.
Sign the certificate? [y/n]:y
-- Select y, the certificate signing and issuing is complete.
1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated

Enable password protection for the private key: If the password protection of the server private key is not disabled, you need to use gs_guc to encrypt the password.

```
gs_guc encrypt -M server -K Gauss@MppDB -D ./
```

After the password is encrypted using gs_guc, two private key password protection files `server.key.cipher` and `server.key.rand` are generated.

**Step 8 Generate the client certificate and private key** - Generate a client private key.

```
oppenssl genrsa -aes256 -out client.key 2048
```

Generate a certificate request file for a client.

```
oppenssl req -config openssl.cnf -new -key client.key -out client.req
```

After the generated certificate request file for client is signed and issued, a formal client certificate `client.crt` is generated.

```
oppenssl ca -config openssl.cnf -in client.req -out client.crt -days 3650 -md sha256
```

**NOTE**

If METHOD is set to `cert` in the `pg_hba.conf` file of the server, the client must use the `username` (common name) configured in the license file (`client.crt`) for the database connection. If METHOD is set to `md5` or `sha256`, the client does not have this restriction.

If password protection for a client private key is not removed, you need to use gs_guc to encrypt the password.

```
gs_guc encrypt -M client -K Gauss@MppDB -D ./
```

After the password is encrypted using gs_guc, two private key password protection, files `client.key.cipher` and `client.key.rand` are generated.

----End

**Replacing Certificates**

Default security certificates and private keys required for SSL connection are configured in LibrA. The formal certificates and keys for the server and client have been obtained from the CA.

**Step 1** Prepare for a certificate and a key. Conventions for configuration file names on the server:
Step 2  Create a compressed package.

Package name: db-cert-replacement.zip

Package format: ZIP

Package file list: server.crt, server.key, server.key.cipher, server.key.rand, client.crt, client.key, client.key.cipher, client.key.rand, cacert.pem. If you need to configure the certificate revocation list (CRL), the list must contain sslcrl-file.crl.

Command:
zip db-cert-replacement.zip client.crt client.key client.key.cipher client.key.rand server.crt server.key server.key.cipher server.key.rand
zip -u ../db-cert-replacement.zip cacert.pem

Step 3  Invoke the certificate replacement interface to replace a certificate. Upload the prepared package db-cert-replacement.zip to any path of a cluster user. For example: /home/gaussdba/test/db-cert-replacement.zip

Run the following command to perform the replacement in coordinator:

```
 gs_om -t cert --cert-file=/home/gaussdba/test/db-cert-replacement.zip
```

Starting SSL cert files replace.

Backing up old SSL cert files.

Backup SSL cert files on BLR1000029898 successfully.
Backup SSL cert files on BLR1000029896 successfully.
Backup SSL cert files on BLR1000029897 successfully.
Backup gds SSL cert files on successfully.
BLR1000029898 replace SSL cert files successfully.
BLR1000029896 replace SSL cert files successfully.
BLR1000029897 replace SSL cert files successfully.
Replace SSL cert files successfully.
Distribute cert files on all coordinators successfully.

You can run the `gs_om -t cert --rollback` command to remotely invoke the interface and `gs_om -t cert --rollback -L`.

-----End

Configuration For Client

Step 1  Run the following command on the client key file:

```
openssl pkcs8 -topk8 -inform PEM -outform DER -in Client.key -out client.pk8
```
Step 2  Copy the client.pk8, client.crt, cacert.pem that were created above to the client machine.

**NOTE**

When the Data Studio tool selects the client SSL key, the key file cannot be selected, and the *.pk8 file needs to be selected. However, the downloaded certificate does not contain the pk8 file.

Step 3  Server Configuration for Client for **Two way** SSL Authentication

<table>
<thead>
<tr>
<th>hostssl</th>
<th>all</th>
<th>all</th>
<th>10.18.158.95/32</th>
<th>cert</th>
</tr>
</thead>
</table>

Server Configuration for Client for **One way** SSL Authentication.

<table>
<thead>
<tr>
<th>hostssl</th>
<th>all</th>
<th>all</th>
<th>10.18.158.95/32</th>
<th>sha256</th>
</tr>
</thead>
</table>

Step 4  During log in to Data Studio password is not validated during Two Way SSL authentication.
SSL password needs to be entered.

----End

4.26 Troubleshooting

1. **Data Studio does not open. What do I do?**

   **Solution:** Check whether Java Runtime Environment (JRE) is missing. Verify the configured Java path in the environment. Refer to [System Requirements](#) for supported Java JDK version.

2. **Data Studio does not open and displays a 'Java Runtime' error when I double-click the Data Studio.exe file. What do I do?**

   **Solution:**
   - For no JRE:
     
     ![Image of Data Studio error message]

     Check whether the Java Runtime Environment (JRE) or Java Development Kit (JDK) version 1.8.0_141 or above with appropriate bit number is installed on the system and Java Home path is set. If there are more than one version of Java installed, then set the -vm parameter in the configuration file. Refer to the [Installing and Configuring Data Studio](#) section to set this parameter. This is a prerequisite for running Data Studio.
     
     - For older versions of JRE:
       ![Image of Data Studio error message]

       Check the version of Java Runtime Environment (JRE) or Java Development Kit (JDK) that is installed on the system. An older version installed on the system causes this error. Update the JRE to version 1.8.0_141 or above with appropriate bit number.

   - **Java Incompatibility:**
Check the version of Java Runtime Environment (JRE) or Java Development Kit (JDK) that is installed on the system. Incompatible Java
bit version installed on the system causes this error. Update the JRE version to 1.8.0_141 or above with appropriate bit number.

It is recommended that run the batch file to check compatibility and launch Data Studio. Refer to Getting Started for more information.

3. **While running the StartDataStudio.bat file the following message displays. What do I do?**

   **Solution:**

<table>
<thead>
<tr>
<th>Message</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>You are attempting to run 32-bit Data Studio on:</td>
<td>Install Java 1.8 32-bit</td>
</tr>
<tr>
<td>● 64 bit OS</td>
<td></td>
</tr>
<tr>
<td>● Microsoft Windows 7 Professional</td>
<td></td>
</tr>
<tr>
<td>● Java 1.8 64-bit JDK (Incompatible)</td>
<td></td>
</tr>
<tr>
<td>Install Java 1.8 32-bit</td>
<td></td>
</tr>
<tr>
<td>Data Studio is supported with minimum Java Version of 1.8</td>
<td>Install Java version 1.8 with appropriate bit number</td>
</tr>
<tr>
<td>Install Java version 1.8 in order to use Data Studio</td>
<td></td>
</tr>
<tr>
<td>You are attempting to run 64-bit Data Studio on:</td>
<td>Install Java 1.8 64-bit</td>
</tr>
<tr>
<td>● 64 bit OS</td>
<td></td>
</tr>
<tr>
<td>● Microsoft Windows 7 Professional</td>
<td></td>
</tr>
<tr>
<td>● Java 1.8 32-bit JDK (Incompatible)</td>
<td></td>
</tr>
<tr>
<td>Install Java 1.8 64-bit</td>
<td></td>
</tr>
<tr>
<td>You are attempting to run 64-bit Data Studio on:</td>
<td>Install 32-bit Data Studio</td>
</tr>
<tr>
<td>● 32 bit OS</td>
<td></td>
</tr>
<tr>
<td>● Microsoft Windows 7 Professional</td>
<td></td>
</tr>
<tr>
<td>● Java 1.8 32-bit JDK (Incompatible)</td>
<td></td>
</tr>
<tr>
<td>Install 32-bit Data Studio</td>
<td></td>
</tr>
</tbody>
</table>

4. **Why does Data Studio not connect to the server even with all valid inputs?**

   **Solution:** Check whether the server is running in the specified IP and port. Check for availability of the specified user by connecting through `gsql`.

5. **What do I do for connection issues while using Data Studio?**

   **Solution:** A connection issue that may occur while using Data Studio is explained with an example:

   Establish a database connection.
   Run the query.
When a connection exception occurs in any one of the database (PostgreSQL), the connection is closed. When the database connection is closed, all the function and procedure tabs, if open, will be closed too.

An error will be seen and the **Object Browser** will show the status of the database.

---

**NOTE**

Only the current database will be disconnected. Other databases will remain connected and re-connection is possible.

Re-connect to the database to proceed with execution.

6. **While fetching a function/procedure containing Chinese comments through a Java application, the Chinese characters are not visible. What do I do?**

   **Solution:** Set **Preferences > Session Setting > Data Studio Encoding** and **File Encoding** to **GBK**, so that Chinese characters are displayed properly.

7. **Connecting to large database, loading large number of queries into SQL Terminal, may result into 'Out of Memory' error or 'Java Heap Space' error. What is the solution?**

   **Solution:** 'Out of Memory' error or 'Java Heap Space' error occurs when Data Studio has used up the maximum allocated Java memory. By default, the configuration file Data Studio.ini (located in the Data Studio install path) contains the entry `-Xmx1200m`, where 1200m denotes 1200 MB as the maximum Java memory that can be used by Data Studio. The memory usage is based on the size of data fetched by Data Studio.

   To resolve this issue, increase the Java memory size to the desired value. For example, update `-Xmx1200m` to `-Xmx2000m` and restart Data Studio. If the updated memory size is used up as well, the same issue might reoccur.

   ---

   **NOTE**

   - As an example for 32-bit Data Studio and 8 GB RAM the value of the Xmx parameter must not cross 2044 MB and for 64-bit Data Studio and 8 GB RAM the value of the Xmx parameter must not cross 6000 MB. The limit may vary based on user's current memory usage.

   For example:
   
   -Xms1024m
   -Xmx1800m

   - The maximum file size that Data Studio can support in the SQL Terminal is based on the value of the Xmx parameter in the Data Studio.ini file and available memory.

8. **While executing an SQL query which returns a large amount of data, Data Studio displays an 'Insufficient Memory' error. What do I do?**

   **Solution:** Data Studio will disconnect the connection profile. Re-establish the connection and continue execution.

9. **While exporting DDL or data why do I get export failed message?**

   **Solution:** This could happen due to any of the following reasons:

   - Invalid file for Client SSL Certificate and/or Client SSL Key have been selected. Select the correct file and try again. Refer to **Adding a Connection** on establishing connection.
10. While performing Show DDL operation why do I get show DDL failed message?
   Solution: This could happen due to any of the following reasons:
   - Invalid file for Client SSL Certificate and/or Client SSL Key have been selected. Select the correct file and try again. Refer to Adding a Connection on establishing connection.
   - Identity of the object in the database could have been changed. Check if the identity of the object has been changed and try again.
   - You may have insufficient privileges. Contact the database administrator to obtain appropriate privileges.

11. While performing Show DDL or Export DDL operation, why do I get this error message?
    "Failed to start the program, because MSVCRT100.dll is missing. Try reinstalling the program to solve the problem?"
    Solution: This is because the operation requires gs_dump.exe to execute, and it requires the Microsoft VC runtime library msvcrt100.dll.
    To solve this, copy the msvcrt100.dll file from the \Windows\System32 folder to \Windows\SysWOW64 folder.

12. Why does the saved connection profile details not show when I try to establish a connection?
    Solution: This could happen if the Profile folder under User Data folder is not available or modified manually. Make sure the Profile folder is present with correct naming convention.

13. When I close and reopen Data Studio the SQL query history information is lost. Why does this happen?
    Solution: This could happen if the Profile folder is missing under User Data folder or the folder has been modified. Make sure the Profile folder is present with correct naming convention.

14. When I try to modify any preference error saving preference message displays. Why do I get this error message?
    Solution: This can happen if the Preferences folder is not present or renamed. Restart Data Studio to resolve this issue.

15. What do I do when Data Studio becomes idle and Data Studio.log file states "No more handles"?
    Solution: Restart Data Studio.

16. What happens if an error occurs after I have edited a table and I am unable to make further changes?
    Solution: All previously edited data will be lost. Close the Edit Table Data window and perform the changes again.

17. Why do I keep getting the message "The number of pasted cell and the selected cell does not match" even though I have made the correct number of cell selection?
A: This can happen if the settings in Preferences > Query Results is set to include column header. The selected cells includes the column header cells as well. Modify the settings to disable include column header option and try again.

18. Why am I not able to edit the temp table with Reuse Connection option off?
A: Turning off Reuse Connection option creates new session. Created temp tables are available for current session only. Turn on Reuse Connection option to edit temporary tables. Refer to Managing SQL Terminal Connections section for more information.

19. What happens if same column is added more than once in Multi-Column sort pop-up?
A: When same column is added more than once in multi-sort pop up table, and Apply is clicked, a notification is displayed. You need to click OK and select the correct column to sort.

![Multi-Column Sort](image)

20. What happens if a column name is not selected in at least one of the sort criteria and Apply is clicked?
A: Following notification is displayed. Once you select a valid column name and click Apply again, this notification is not displayed.

![Multi-column Sort Error](image)

21. What happens when you click on cancel when multiple create table queries are running in SQL terminal window.
A: Canceling queries might throw an error in console displaying the table name that is not created. In that case it is recommended to drop that particular table in order to perform operation on a table with the same name.

22. When the user is not able to logon to DS due to security keys are compromised?

**Solution:** Follow the steps to generate the new security keys.

a. Delete the security folder from *Datastudio folder > Userdata > Security folder*.

b. Restart Data Studio.

c. New security folder is created and the keys will be regenerated.

d. Saved password will be lost and user should re-enter the password to logon to Data Studio.

### 4.27 FAQs

1. **What aspects must be checked in case of a connection failure?**
   
   A: Check the following:
   
   - Verify the **Connection Properties**, to check whether the input to the connection properties is correct.
   
   - Check whether the server and client versions are compatible.

   - Check whether *database/pg_hba.conf* file is configured properly. Refer to the server manual for more details.

   - Check whether *Data Studio.ini* file is configured properly.

2. **When I try to establish a second connection with a different server using the same SSL certificates, why is the connection successful?**

   A: If the same SSL certificates are used by different server, then the second connection will be successful because the certificates are cached.

   When you try to establish a second connection with a different server using different SSL certificates, the connection will fail because of a certificate mismatch.

3. **When I right-click on the function/procedure and perform 'Refresh' in the Object Browser, the function/procedure is not visible. What can be the reason?**

   A: This may happen when you drop a function/procedure and recreate it. In this case, refresh the parent folder to view the function/procedure in the Object Browser.

4. **What action must be taken if a critical error occurs in a database session and is unable to proceed?**

   A: Critical error can occur in some of the following cases. Check whether:

   - The connection is left idle for long time and has timed out.

   - The server is running or not.

   - There is enough memory available on the server and no "out of memory" is reported on the server.

5. **What is a constraint?**
A: Constraints are used to restrict the insertion of unwanted data in columns. You can create constraints on single or multiple columns of any table. It maintains the data integrity of the table.

There are three types of constraints supported:
- Primary Key constraint
- Unique Key constraint
- Check constraint

6. **What is an index?**
   A: An index is a copy of select columns of data from a table that can be searched very efficiently. It also includes a low level disk block address or a direct link to the complete row of data it was copied from.

7. **What is the default encoding for Data Studio’s files?**
   A: Exported, imported, and system files are encoded with the system’s default encoding as configured in Settings > Preferences. The default encoding is UTF-8.

8. **When I try to open another instance of Data Studio multiple instances of Data Studio is not supported message displays. Why do I get this error message?**
   A: Opening multiple instances of Data Studio by the same user is not supported.

9. **When I try to perform DDL operation on an object, the task keeps running indefinitely and I am unable to cancel the task. What could be the reason?**
   A: This can happen if there is another active DML/DDL operation being performed on the same object. Close all active DML/DDL operations on the object and try again. If the problem still persists, it could be that another user might be performing a DML/DDL operation on that object. Try after sometime.

10. **Why is the exported query result different from the data available in the Results tab?**
    A: For export result set data, the query is re-executed using a new connection. Hence the exported result may differ from the data shown in the Results tab.

11. **Why does last login information show “Last login details not available”?**
    A: Last login details not available is shown, when you are connected to the older version of the database server or you have logged in to the database for the first time after the database has been created.

12. **Why is the error marked incorrectly in the SQL Terminal?**
    A: This happens when server returns the incorrect line number. Review the error message in Messages tab and navigate to the corresponding line number to fix the error.

13. **Do Show DDL and Export DDL display dropped column information?**
    A: Yes, Show DDL and Export DDL operation will display the dropped column information.

14. **Why does Data Studio not launch after I have modified the -Xmx parameter?**
    A: This happens if the value defined for -Xmx parameter may be invalid. Refer to Installing and Configuring Data Studio.
15. **How can I access a Terminal quicker if I have opened multiple number of Terminals or tab?**
   
   **A:** After the number of opened Terminals or tabs reaches a certain limit based on screen resolution an icon (③) displays with a drop-down option at the end of the Terminal list. Click the ③ icon and select the required Terminal from the drop-down list. If the ③ is not available, then use the tooltip to identify the Terminal or tab. Terminal name can be searched by typing the search value above the list of SQL Terminal names.

   **Example:**
   - "s," this displays all Terminal name that starts with s.
   - "test," this displays all Terminal name that starts with test.
   - "2," this displays all Terminal name that contains 2 in them.

16. **Why is that after I change the language DS restarts but the language does not change?**
   
   **A:** Sometimes the language may not reflect the selected change post restart. Manually restart DS to open the tool in selected language.

17. **Why does the last login details information not display?**
   
   **A:** At times the server returns an error while trying to fetch last login details. In such scenarios the last login pop-up message does not display.

18. **When viewing/exporting DDL, why does the Chinese text not show properly?**
   
   **A:** This happens if the SQL, DDL, object names or data contains Chinese text and the Data Studio file encoding is not set to GBK. To solve this, go to Settings > Preferences > Environment > File Encoding Preferences and set the encoding to GBK.

   The supported combinations of Database and Data Studio encoding for export operation are shown in Table **1 Supported combinations of file encoding**.

   **To open/view the exported files in Windows Explorer:** Files exported with UTF-8 encoding can be opened/viewed by double-clicking it or by right-clicking on the file and selecting Open. Files exported with GBK encoding must be opened in MS-Excel® using the import external data feature (Data > Get External Data > From Text).

### Table 4-26 Supported combinations of file encoding used in the Database and Data Studio

<table>
<thead>
<tr>
<th>Database Encoding</th>
<th>Data Studio File Encoding</th>
<th>Support for Chinese Text in Table Names</th>
<th>Support for English Text in Table Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>GBK</td>
<td>GBK</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GBK</td>
<td>UTF-8</td>
<td>No - Incorrect details</td>
<td>No - Incorrect details</td>
</tr>
<tr>
<td>UTF-8</td>
<td>GBK</td>
<td>No - Export Fails</td>
<td>No - Incorrect details</td>
</tr>
<tr>
<td>Database Encoding</td>
<td>Data Studio File Encoding</td>
<td>Support for Chinese Text in Table Names</td>
<td>Support for English Text in Table Names</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------</td>
<td>----------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>UTF-8</td>
<td>UTF-8</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UTF-8</td>
<td>LATIN1</td>
<td>No - Export Fails</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL_ASCII</td>
<td>GBK</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL_ASCII</td>
<td>UTF-8</td>
<td>No - Incorrect details</td>
<td>No - Incorrect details</td>
</tr>
</tbody>
</table>

19. **Why do I get the error message "Conversion between GBK and LATIN1 is not supported"?**

   **A:** This message occurs if the Data Studio and Database encoding selected are incompatible. To solve this, select the compatible encoding. Compatible encoding is shown in **Table 4-27**

**Table 4-27 Compatible Encoding Formats**

<table>
<thead>
<tr>
<th>Data Studio Encoding</th>
<th>Database Encoding</th>
<th>Compatible</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTF-8</td>
<td>GBK</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>LATIN1</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>SQL_ASCII</td>
<td>Yes</td>
</tr>
<tr>
<td>GBK</td>
<td>UTF-8</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>LATIN1</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>SQL_ASCII</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL_ASCII</td>
<td>UTF-8</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>LATIN1</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>GBK</td>
<td>Yes</td>
</tr>
</tbody>
</table>

20. **Why is the PL/SQL procedure I compiled and executed is saved as PL/SQL function?**

   **A:** The database does not differentiate between PL/SQL function and procedure. According to the database all procedures are functions. Hence PL/SQL procedure is saved as PL/SQL function.

21. **Why is that I am not able to edit the distribution key?**

   **A:** The database allows you to edit the distribution key only for the first insert operation.

22. **While editing table data if I do not enter a value for default value column, will the value be added by the database server?**
A: Yes, the database server add the value but the value will not be visible after save in the Edit Table Data tab. Use the refresh option from the Edit Table Data tab or re-open the table again to view the added default value (s).

23. While modifying/deleting table data why do I get a pop-up stating that more than one matching row found?
A: This happens because there are additional rows detected for modification/deletion based on Custom Unique Key or All Columns selection. If Custom Unique Key is selected, then it will delete/modify the rows that have exact match of the data in the column selected for deletion/modification. If All Columns is selected, then it will delete/modify the rows that match data in all columns. Hence this duplicate records matching the Custom Unique Key or All Columns will be deleted/modified if Yes is selected. If No is selected, the row that is not saved will be marked for correction.

24. When I right-click on a text box I see additional context menu options. Why does this happen?
A: The additional context menu options like Right to left Reading order, Show Unicode control characters and so on are provided by Windows 7 in case the keyboard you are using supports right to left and left to right input.

25. What are the objects that are not supported for batch export DDL & DDL and Data operations?
A: Following objects are not supported for DDL & DDL and Data operations.

Export DDL:
- Connection
- Database
- Tablespace
- Foreign Table
- Sequence
- Column
- Index
- Constraint
- Partition
- Function/Procedure Group
- Regular Tables Group
- Views Group
- Schemas Group
- Catalogs Group

Export DDL and Data:
- Connection
- Database
- Tablespace
- Foreign Table
- Sequence
- Column
- Index
26. Will the queries in SQL Terminal commit if the resultset is modified and saved with Reuse Connection On and Auto Commit Off?

A: No. Queries will only be committed when COMMIT command is executed in the Terminal.

<table>
<thead>
<tr>
<th>Auto Commit</th>
<th>Reuse Connection</th>
<th>Resultset Save</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>On</td>
<td>Commit</td>
</tr>
<tr>
<td>On</td>
<td>Off</td>
<td>Commit</td>
</tr>
<tr>
<td>Off</td>
<td>On</td>
<td>Does not commit</td>
</tr>
<tr>
<td>Off</td>
<td>Off</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

27. When I query a temp table from a new SQL Terminal the resultset displays incorrect table details. Why does this happen?

A: When you query a temp table from a new SQL Terminal or with the Reuse Connection Off, the resultset displays information of a regular/partition/foreign table, if a table with the same name as the temp table exists.

**NOTE**

If the Reuse Connection is On, the resultset displays information of the temp table even if another table with the same name exists.

28. Which are the operations that are performed on a locked object does not run in the background but needs to be manually closed?

A: Following are the operations that do not run in background while the object is locked in another operation:

<table>
<thead>
<tr>
<th>Operations</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Renaming table</td>
<td>Creating constraint</td>
</tr>
<tr>
<td>Setting schema on table</td>
<td>Creating index</td>
</tr>
<tr>
<td>Setting tablespace in table</td>
<td>Renaming schema</td>
</tr>
<tr>
<td>Setting description in table</td>
<td>Adding column</td>
</tr>
<tr>
<td>Renaming partition</td>
<td>-</td>
</tr>
</tbody>
</table>

29. Do we have a limit on the column and row size while exporting table data to excel?
A: Yes, xlsx format supports maximum of 1 million rows and 16384 columns and xls format supports maximum of 64K rows and 256 columns.
5 GDS - Parallel Data Loader

5.1 Installing, Configuring, and Starting GDS

Scenarios

DWS uses GDS to allocate the source data for parallel data import. Deploy GDS on the data server.

If a large volume of data is stored on multiple servers, deploy, configure, and start GDS on each server. Then, data on all the servers can be imported in parallel. The procedure of installing, configuring, and starting GDS is the same on each data server. This section describes how to perform this procedure on one data server.

Context

- GDS can be installed in the following OSs:

  x86:
  - SUSE Linux Enterprise Server 10 SP4 x86_64
  - SUSE Linux Enterprise Server 11 SP1/SP2/SP3/SP4 x86_64
  - SUSE Linux Enterprise Server 12 SP0/SP1/SP2/SP3 x86_64
  - Red Hat Enterprise Linux Server release 6.4/6.5/6.6/6.7/6.8/6.9/7.0/7.1/7.2/7.3/7.4 x86_64
  - Community Enterprise Operating System 6.4/6.5/6.6/6.7/6.8/6.9/7.0/7.1/7.2/7.3/7.4 x86_64

- The GDS version must match the cluster version. For example, GDS V100R008C00 matches DWS 1.3.X. Otherwise, the import or export may fail, or the import or export process may fail to respond.

Therefore, do not use the GDS of an earlier version. After the database is upgraded, download the GDS of the new version as instructed in Procedure. When the import or export starts, DWS checks the GDS version. If it does not match the DWS version, DWS displays an error message and terminates the import or export.
To obtain the version number of GDS, run the following command in the GDS decompression directory:

```
gds -V
```

To view the database version, run the following SQL statement after connecting to the database:

```
SELECT version();
```

**Procedure**

**Step 1** Prepare a GDS server and download the GDS package and SSL certificate. For details, see "Using GDS to Import and Export Data" in *Data Warehouse Service User Guide*.

**Step 2** Log in as user `root` to the data server where GDS is to be installed and run the following command to create the directory for storing the GDS package:

```
mkdir -p /opt/bin
```

**Step 3** Upload the GDS package to the created directory.

Use the SUSE Linux package as an example. Upload the GDS package `GaussDB-Kernel-V300R002C00-SUSE11-64bit-Gds.tar.gz` to the directory created in the previous step.

**Step 4** (Optional) If SSL is used, upload the SSL certificate to the directory created in **Step 2**.

**Step 5** Go to the new directory and decompress the package.

```
cd /opt/bin
tar -zxvf GaussDB-Kernel-V300R002C00-SUSE11-64bit-Gds.tar.gz
```

**Step 6** Create a GDS user and the user group to which the user belongs. This user is used to start GDS and read source data.

```
groupadd gdsgrp
useradd -g gdsgrp gds_user
```

**Step 7** Change the owner of the GDS package directory and source data file directory to the GDS user.

```
chown -R gds_user:gdsgrp /opt/bin/gds
chown -R gds_user:gdsgrp /input_data
```

**Step 8** Switch to user `gds_user`.

```
su - gds_user
```

**Step 9** Start GDS.

GDS is green software and can be started after being decompressed. You can start GDS by running the `gds` command, or modifying the `gds.conf` configuration file and then running the `gds_ctl.py` command. The `gds` command is recommended when you do not need to import data again. The `gds.conf` configuration file is recommended when you need to import data again.

- Run the `gds` command to start GDS.
  - If data is transmitted in non-SSL mode, run the following command to start GDS:
    
    ```
gds -d dir -p ip:port -H address_string -l log_file -D -t worker_num
    ```
    
    Example:
If data is transmitted in SSL mode, run the following command to start GDS:
```bash
gds -d dir -p ip:port -H address_string -l log_file -D -t worker_num --enable-ssl --ssl-dir Cert_file
```

Example:
Run the following command to upload the SSL certificate mentioned in Step 4 to /opt/bin:
```bash
gds -d dir -p ip:port -H address_string -l log_file -D --enable-ssl --ssl-dir Cert_file
```

Replace the information in italic as required.

- **-d dir**: directory storing data files that contain data to be imported. It is `/input_data/` in this tutorial.
- **-p ip:port**: listening IP address and port for GDS. The default value is `127.0.0.1:8098`. Replace it with the IP address of a 10GE network that can communicate with DWS. The listening port can be any one ranging from 1024 to 65535. The default port is **8098**. This parameter is set to `192.168.0.90:5000` in this tutorial.
- **-H address_string**: network segment for hosts that can connect to and use GDS. The value must be in CIDR format. Set this parameter to enable DWS to access GDS for data import. Ensure that the network segment covers all hosts in DWS.
- **-l log_file**: GDS log directory and log file name. It is `/opt/bin/gds/gds_log.txt` in this tutorial.
- **-D**: GDS in daemon mode. This parameter is used only in Linux.
- **-t worker_num**: number of concurrent GDS threads. If the data server and DWS have available I/O resources, you can increase the number of concurrent GDS threads.

GDS determines the number of threads based on the number of concurrent import transactions. That is, even if multi-thread import is configured before GDS startup, the import of a single transaction will not be accelerated. By default, an **INSERT** statement is an import transaction.

- **--enable-ssl**: enables SSL for data transmission.
- **--ssl-dir Cert_file**: SSL certificate directory. Set it to the certificate directory mentioned in Step 4.

- For details about GDS parameters, see "Parameter Description".

**Run the gds_ctl.py command to start GDS.**

a. Run the following command to go to the **config** directory of the GDS package and modify the **gds.conf** configuration file. For details about the parameters in the **gds.conf** configuration file, see **Table 5-1**.

```bash
vim /opt/bin/gds/config/gds.conf
```

For example:

The gds.conf configuration file contains the following information:

```xml
<?xml version="1.0"?>
<config>
<gds name="gds1" ip="192.168.0.90" port="5000" data_dir="/input_data/" err_dir="/err" data_seg="100MB" err_seg="100MB" log_file="/log/gds_log.txt" host="10.10.0.1/24" daemon='true' recursive='true' parallel="32"></gds>
</config>
```
Information in the configuration file is described as follows:

- The data server IP address is **192.168.0.90** and the GDS listening port is **5000**.
- Data files are stored in the `/input_data/` directory.
- Error log files are stored in the `/err` directory.
- The size of a single data file is 100 MB.
- The size of a single error log file is 100 MB.
- Logs are stored in the `/log/gds_log.txt` file.
- Only nodes with the IP address being 10.10.0.* can be connected.
- The GDS process is running in daemon mode.
- Recursive data file directories are used.
- The number of concurrent import threads is 2.

**b. Start GDS and check whether it has been started.**

```
python gds_ctl.py start
```

Example:

```
cd /opt/bin/gds
python gds_ctl.py start
```

**gds.conf Parameter Description**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Identifier</td>
<td>-</td>
</tr>
<tr>
<td>ip</td>
<td>Listening IP address</td>
<td>The IP address must be valid. Default value: <strong>127.0.0.1</strong></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Value Range</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| port      | Listening port | Value range: an integer ranging from 1024 to 65535  
  Default value: **8098** |
| data_dir  | Data file directory | - |
| err_dir   | Error log file directory | Default value: data file directory |
| log_file  | Log file Path | - |
| host      | Hosts that can be connected to GDS. The value must in CIDR format and this parameter is set for the Linux OS only. | - |
| recursive | Whether the data file directories are recursive | Value range:  
  ● **true**: recursive  
  ● **false**: not recursive  
  Default value: **false** |
| daemon    | Whether the process is running in daemon mode | Value range:  
  ● **true**: The process is running in daemon mode.  
  ● **false**: The process is not running in daemon mode.  
  Default value: **false** |
| parallel  | Number of concurrent data import threads | Value range: 0 to 32 (integer)  
  Default value: **1** |

### 5.2 Stopping GDS

**Scenarios**

Stop GDS after data is imported successfully.

**Procedure**

**Step 1** Log in as user **gds_user** to the data server where GDS is installed.

**Step 2** Select the mode of stopping GDS based on the mode of starting it.

- If GDS is started using the **gds** command, perform the following operations to stop GDS:
  
a. Query the GDS process ID:

```bash
ps -ef|grep gds
```

For example, the GDS process ID is 128954.
b. Run the `kill` command to stop GDS. **128954** in the command is the GDS process ID.

```
kill -9 128954
```

- If GDS is started using the `gds_ctl.py` command, perform the following operations to stop GDS:

```
cd /opt/bin/gds
python gds_ctl.py stop
```

```
----End
```

## 5.3 Example

### Example: Parallel Import from Multiple Data Servers

The data servers reside on the same intranet as the cluster. Their IP addresses are 192.168.0.90 and 192.168.0.91. Source data files are in CSV format.

1. Create the target table `tpcds.reasons`.

```
CREATE TABLE tpcds.reasons
(
  r_reason_sk integer not null,
  r_reason_id char(16) not null,
  r_reason_desc char(100)
);
```

2. Log in to each GDS data server as user `root` and create the `/input_data` directory for storing data files on the servers. The following takes the data server whose IP address is 192.168.0.90 as an example. Operations on the other server are the same.

```
mkdir -p /input_data
```

3. (Optional) Create a user and the user group it belongs to. The user is used to start GDS. If the user and user group already exist, skip this step.

```
groupadd gdsgrp
useradd -g gdsgrp gds_user
```

4. Evenly distribute source data files to the `/input_data` directories on the data servers.

5. Change the owners of source data files and the `/input_data` directory on each data server to `gds_user`. The data server with the IP address 192.168.0.90 is used as an example.

```
chown -R gds_user:gdsgrp /input_data
```

6. Log in to each data server as user `gds_user` and start GDS.

   The GDS installation path is `/opt/bin/gds`. Source data files are stored in `/input_data/`. The IP addresses of the data servers are 192.168.0.90 and 192.168.0.91. The GDS listening port is 5000. GDS runs in daemon mode.

   Start GDS on the data server whose IP address is 192.168.0.90.

```
/gds/gds -d /input_data -p 192.168.0.90:5000 -H 10.10.0.1/24 -D
```

   Start GDS on the data server whose IP address is 192.168.0.91.

```
/gds/gds -d /input_data -p 192.168.0.91:5000 -H 10.10.0.1/24 -D
```

7. Create the foreign table `tpcds.foreign_tpcds_reasons` for the source data. Set import mode parameters as follows:
- Set the import mode to **Normal**.
- When GDS is started, the source data file directory is `/input_data` and the GDS listening port is 5000. Therefore, set `location` to `gsfs://192.168.0.90:5000/* | gsfs://192.168.0.91:5000/*`.

Information about the data format is set based on data format parameters specified during data export. The parameter settings are as follows:
- `format` is set to `CSV`.
- `encoding` is set to `UTF-8`.
- `delimiter` is set to `E'\x08'`.
- `quote` is set to `0x1b`.
- `null` is set to an empty string without quotation marks.
- `escape` is set to a double quotation mark.
- `header` is set to `false`, indicating that the first row is regarded as a data row when a file is imported.

Set import error tolerance parameters as follows:
- Set `PER NODE REJECT LIMIT` (number of allowed data format errors) to `unlimited`. In this case, all the data format errors detected during data import will be tolerated.
- Set `LOG INTO` to `err_tpcds_reasons`. The data format errors detected during data import will be recorded in the `err_tpcds_reasons` table.

Based on the above settings, the foreign table is created using the following statement:

```sql
CREATE FOREIGN TABLE tpcds.foreign_tpcds_reasons
(
    r_reason_sk integer not null,
    r_reason_id char(16) not null,
    r_reason_desc char(100)
) SERVER gsmpp_server OPTIONS (location 'gsfs://192.168.0.90:5000/* | gsfs://192.168.0.91:5000/*', format 'CSV', mode 'Normal', encoding 'utf8', delimiter E'\x08', quote E'\x1b', null '', fill_missing_fields 'false') LOG INTO err_tpcds_reasons PER NODE REJECT LIMIT 'unlimited';
```

8. Import data through the foreign table `tpcds.foreign_tpcds_reasons` to the target table `tpcds.reasons`.

```
INSERT INTO tpcds.reasons SELECT * FROM tpcds.foreign_tpcds_reasons;
```

9. Query data import errors in the `err_tpcds_reasons` table and rectify the errors (if any). For details, see "Handling Error Tables" in *Data Warehouse Service Database Developer Guide*.

```sql
SELECT * FROM err_tpcds_reasons;
```

10. After data import is complete, log in to each data server as user `gds_user` and stop GDS.

The data server with the IP address 192.168.0.90 is used as an example. The GDS process ID is 128954.

```
ps -ef|grep gds
```

```
gds_user 128954  1  0 15:03 ? 00:00.00 gds -d /input_data -p 192.168.0.90:5000 -D
gds_user 129003 118723  0 15:04 pts/0 00:00.00 grep gds
kill -9 128954
```

---

**Example: Data Import Using Multiple Threads**

The data server resides on the same intranet as the cluster. The server IP address is 192.168.0.90. Source data files are in CSV format. Data will be imported to two tables using multiple threads in **Normal** mode.
1. In the database, create the target tables `tpcds.reasons1` and `tpcds.reasons2`.

   ```sql
   CREATE TABLE tpcds.reasons1
   (    
       r_reason_sk integer not null,
       r_reason_id char(16) not null,
       r_reason_desc char(100)
   ) ;
   CREATE TABLE tpcds.reasons2
   (    
       r_reason_sk integer not null,
       r_reason_id char(16) not null,
       r_reason_desc char(100)
   ) ;
   ```

2. Log in to the GDS data server as user `root`, and then create the data file directory `/input_data` and its sub-directories `/input_data/import1/` and `/input_data/import2/`.

   ```bash
   mkdir -p /input_data
   ```

3. Store the source data files of the target table `tpcds.reasons1` in `/input_data/import1/` and the source data files of the target table `tpcds.reasons2` in `/input_data/import2/`.

4. (Optional) Create a user and the user group it belongs to. The user is used to start GDS. If the user and user group already exist, skip this step.

   ```bash
   groupadd gdsgrp
   useradd -g gdsgrp gds_user
   ```

5. Change the owners of source data files and the `/input_data` directory on the data server to `gds_user`.

   ```bash
   chown -R gds_user:gdsgrp /input_data
   ```

6. Log in to the data server as user `gds_user` and start GDS.

   ```bash
   The GDS installation path is `/gds`. Source data files are stored in `/input_data/`. The IP address of the data server is 192.168.0.90. The GDS listening port is 5000. GDS runs in daemon mode. The degree of parallelism is 2. A recursive directory is specified.

   ```bash
   /gds/gds -d /input_data -p 192.168.0.90:5000 -H 10.10.0.1/24 -D -t 2 -r
   ```

7. In the database, create the foreign tables `tpcds.foreign_tpcds_reasons1` and `tpcds.foreign_tpcds_reasons2` for the source data.

   The foreign table `tpcds.foreign_tpcds_reasons1` is used as an example to describe how to set parameters in a foreign table.

   Set import mode parameters as follows:
   - Set the import mode to `Normal`.
   - When GDS is started, the configured source data file directory is `/input_data` and the GDS listening port is 5000. However, source data files are actually stored in `/input_data/import1/`. Therefore, set `location` to `gsfs://192.168.0.90:5000/import1/*`.

   Information about the data format is set based on data format parameters specified during data export. The parameter settings are as follows:
   - `format` is set to `CSV`.
   - `encoding` is set to `UTF-8`.
   - `delimiter` is set to `E'\x08'`.
   - `quote` is set to `0x1b`.
   - `null` is set to an empty string without quotation marks.
   - `escape` is set to a double quotation mark.
- header is set to false, indicating that the first row is regarded as a data row when a file is imported.

Set import error tolerance parameters as follows:
- Set PER NODE REJECT LIMIT (number of allowed data format errors) to unlimited. In this case, all the data format errors detected during data import will be tolerated.
- Set LOG INTO to err_tpcds_reasons1. The data format errors detected during data import will be recorded in the err_tpcds_reasons1 table.
- If the last column (fill_missing_fields) in a source data file is missing, the NULL column will be automatically added to the target file.

Based on the above settings, the foreign table tpcds.foreign_tpcds_reasons1 is created using the following statement:

```
CREATE FOREIGN TABLE tpcds.foreign_tpcds_reasons1
(
  r_reason_sk integer not null,
  r_reason_id char(16) not null,
  r_reason_desc char(100)
) SERVER gsmpp_server OPTIONS (location 'gsfs://192.168.0.90:5000/import1/*', format 'CSV', mode 'Normal', encoding 'utf8', delimiter E'\x08', quote E'\x1b', null '', fill_missing_fields 'on') LOG INTO err_tpcds_reasons1 PER NODE REJECT LIMIT 'unlimited';
```

Based on the above settings, the foreign table tpcds.foreign_tpcds_reasons2 is created using the following statement:

```
CREATE FOREIGN TABLE tpcds.foreign_tpcds_reasons2
(
  r_reason_sk integer not null,
  r_reason_id char(16) not null,
  r_reason_desc char(100)
) SERVER gsmpp_server OPTIONS (location 'gsfs://192.168.0.90:5000/import2/*', format 'CSV', mode 'Normal', encoding 'utf8', delimiter E'\x08', quote E'\x1b', null '', fill_missing_fields 'on') LOG INTO err_tpcds_reasons2 PER NODE REJECT LIMIT 'unlimited';
```

8. Import data through the foreign table tpcds.foreign_tpcds_reasons1 to tpcds.reasons1 and through tpcds.foreign_tpcds_reasons2 to tpcds.reasons2.

```
INSERT INTO tpcds.reasons1 SELECT * FROM tpcds.foreign_tpcds_reasons1;
INSERT INTO tpcds.reasons2 SELECT * FROM tpcds.foreign_tpcds_reasons2;
```

9. Query data import errors in the err_tpcds_reasons1 and err_tpcds_reasons2 tables and rectify the errors (if any). For details, see "Handling Error Tables" in Data Warehouse Service Database Developer Guide.

```
SELECT * FROM err_tpcds_reasons1;
SELECT * FROM err_tpcds_reasons2;
```

10. After data import is complete, log in to the data server as user gds_user and stop GDS.

The GDS process ID is 128954.

```
ps -ef | grep gds
gds_user 128954 1 0 15:03 ? 00:00:00 gds -d /input_data -p 192.168.0.90:5000 -D -t 2 -r
gds_user 129003 118723 0 15:04 pts/0 00:00:00 grep gds
kill -9 128954
```

### 5.4 gds Command Introduction

**Context**

gds can be used to import and export data of DWS.
Syntax

gds [ OPTION ] -d DIRECTORY

The -d and -H parameters are mandatory and option is optional. gds provides the file data from DIRECTORY for DWS to access.

Before starting GDS, you need to ensure that your GDS version is consistent with the database version. Otherwise, the database will display an error message and terminate the import and export operations. You can view the specific version through the -V parameter.

Parameter Description

- **-d dir**
  Set the directory of the data file to be imported.

- **-p ip:port**
  Set the IP address and port to be listened to of the GDS.
  Value range of the IP address: The IP address must be valid.
  Default value: **127.0.0.1**
  Value range of the listening port is a positive integer ranging from 1024 to 65535.
  Default value of **PORT**: **8098**

- **-l log_file**
  Set the log file.

- **-H address_string**
  Set the hosts that can be connected to the GDS. This parameter must be the CIDR format and it supports the Linux system only. If multiple network segments need to be configured, use commas (,) to separate them. For example, `-H 10.10.0.0/24, 10.10.5.0/24`.

- **-e dir**
  Set the saving path of error logs generated when data is imported.
  Default value: **data file directory**

- **-E size**
  Set the upper thread of error logs generated when data is imported.
  Value range: 0 < size < 1 TB. The value must be a positive integer plus the unit. The unit can be KB, MB, or GB.

- **-S size**
  Set the upper limit of the exported file size.
  Value range: 1 MB < size < 100 TB. The value must be a positive integer plus the unit. The unit can be KB, MB, or GB. If KB is used, the value must be greater than 1024 KB.

- **-t worker_num**
  Set the number of concurrent imported and exported working threads.
  Value range: The value is a positive integer ranging between 0 and 32 (included).

- **-s status_file**
Sets the status file. This parameter supports the Linux system only.

- **-D**
The GDS is running on the backend and this parameter supports the Linux system only.

- **-r**
Traverse files in the recursion directory and this parameter supports the Linux system only.

- **-h**
Shows help information.

- **--enable-ssl**
Use the SSL authentication mode to communicate with clusters.

- **--ssl-dir** Cert_file
Before using the SSL authentication mode, specify the path for storing the authentication certificates.

- **--debug-level**
Sets the debug log level of the GDS to control the output of GDS debug logs.

**Value range: 0, 1, and 2**

- **0**: Only the file list related to log import and export is printed. If the log volume is small, set the parameter to this value only when the system is at normal state.

- **1**: All the log information is printed, including the connection information, session switch information, and statistics on each node. You are advised to set the parameter to this value only during troubleshooting.

- **2**: Detailed interaction logs and their status are printed to generate a huge number of debug logs to help identify the fault causes. You are advised to set the parameter to this value only during troubleshooting.

**Default value: 0**

**Examples**

Data file is saved in the `/data` directory, the IP address is 192.168.0.90, and the listening port number is 5000.

```
gds -d /data/ -p 192.168.0.90:5000 -H 10.10.0.1/24
```

Data file is saved in the subdirectory of the `/data` directory, the IP address is 192.168.0.90, and the listening port number is 5000.

```
gds -d /data/ -p 192.168.0.90:5000 -H 10.10.0.1/24 -r
```

Data file is saved in the `/data` directory, the IP address is 192.168.0.90, and the listening port number is 5000 which is running on the backend. The log file is saved in the `/log/gds_log.txt` file, and the specified number of the concurrently imported working threads is 32.

```
gds -d /data/ -p 192.168.0.90:5000 -H 10.10.0.1/24 -l /log/gds_log.txt -D -t 32
```

Data file is saved in the `/data` directory, the IP address is 192.168.0.90, and the listening port number is 5000. Only the IP address of **10.10.0.*** can be connected.

```
gds -d /data/ -p 192.168.0.90:5000 -H 10.10.0.1/24
```
Data files are stored in the `/data/` directory, the IP address of the directory is **192.168.0.90**, and the listening port number is **5000**. Only the node whose IP address is **10.10.0.*** can be connected to. The node communicates with the cluster using the SSL authentication mode, and the certificate files are stored in the `/certfiles/` directory.

```
gds -d /data/ -p 192.168.0.90:5000 -H 10.10.0.1/24 --enable-ssl --ssl-dir /certfiles/
```

**NOTE**

- One GDS provides the import and export services for one cluster only at a time.
- For security purpose, specify the IP address and the listening port through `-p`.
- The certificate file includes the root certificate `cacert.pem`, level-2 certificate file `client.crt`, and private key file `client.key`.
- The password protection files `client.key.rand` and `client.key.cipher` are used when the system loading certificates.

### 5.5 gds_ctl.py Command Introduction

**Context**

`gds_ctl.py` can be used to start and stop `gds` if `gds.conf` has been configured.

**Prerequisites**

Run the following commands on Linux OS: You need to ensure that the directory structure is as follows before the execution:

```
|--gds
  |--gds_ctl.py
  |--config
    |--gds.conf
    |--gds.conf.sample
```

or

```
|--gds
  |--gds_ctl.py
  |
  `--gds.conf
  `--gds.conf.sample
```

**Content of `gds.conf`:**

```xml
<?xml version="1.0"?>
<config>
  <gds name="gds1" ip="127.0.0.1" port="8098" data_dir="/data" err_dir="/data" err_seg="100MB" err_dir="1000MB" log_file="/gds.log" host="10.10.0.1/24" daemon='true' recursive='true' parallel="32"></gds>
</config>
```

**Configuration description of `gds.conf`:**
- **name**: tag name
- **ip**: IP addresses to be listened to
- **port**: Port number to be listened to
  
  Value range: an integer ranging from 1024 to 65535
  
  Default value: **8098**

- **data_dir**: data file directory
- **err_dir**: error log file directory
- **log_file**: log file path
- **host**: Hosts that can be connected to the GDS.
- **recursive**: Whether the data file directory is recursive.
  
  Value range:
  - **true**: indicates the recursion data file directory.
  - **false**: indicates the data file directory is not recursive.

- **daemon**: Specifies whether the service is running in DAEMON mode.
  
  Value range:
  - **true**: indicates the server is running in the DAEMON mode.
  - **false**: indicates the server is not running in the DAEMON mode.

- **parallel**: Number of concurrently imported and exported working threads
  
  The default concurrency is 1, and the maximum is 32.

**Syntax**

```bash
gds_ctl.py [ start | stop all | stop [ ip: ] port | stop | status ]
```

**Description**

`gds_ctl.py` can be used to start or stop GDS if `gds.conf` is configured.

**Parameter Description**

- **start**
  
  Enables the GDS configured in `gds.conf`.

- **stop**
  
  Stops the running instance started by the configuration file in the GDS that can be disabled by the current users.

- **stop all**
  
  Stops all the running instances in the GDS that can be disabled by the current users.

- **stop [ ip: ] port**
  
  Stops the specific running GDS instance that can be closed by the current user. If `ip:port` is specified when the GDS is started, stop the corresponding `ip:port` to be specified. If the IP address is not specified when the GDS is started, you need to stop the specified port only. The stop fails if different information is specified when the GDS is started or stopped.

- **status**
  
  Query the running status of the GDS instance started by the `gds.conf`.
Examples

Start the GDS.
```
python gds_ctl.py start
```

Stop the GDS started by the configuration file.
```
python gds_ctl.py stop
```

Stop all the GDS instances that can be stopped by the current user.
```
python gds_ctl.py stop all
```

Stop the GDS instance specified by [ip:]port that can be closed by the current user.
```
python gds_ctl.py stop 127.0.0.1:8098
```

Query the GDS status.
```
python gds_ctl.py status
```
6 Migration Tool - SQL Syntax Migration Tool

6.1 Overview

After switching to DWS databases, you may need to perform migration. Database migration includes user data migration and SQL script migration of applications. The latter one is a complex, risky, and time-consuming process.

Migration Tool is a command line tool running on Linux or Windows. It is designed to provide simple, fast, and reliable SQL script migration of applications. Migration Tool parses the SQL scripts of source database applications through its syntax migration logic and migrates the scripts to the ones applicable to DWS databases.

Migration Tool does not require a connection to databases, and performs migration offline. The tool also displays the status of a migration process and logs the errors that occur during the process, helping quickly locate faults.

Migration Objects

Migration Tool can migrate the following Teradata and Oracle database objects:

- General objects: SQL schemas and SQL queries
- Objects supported only by Oracle: PL/SQL objects
- Objects supported only by Teradata: Perl files containing BTEQ and SQL_LANG scripts

Migration Process

The process of migrating SQL scripts by using Migration Tool is as follows:

1. Export the SQL scripts from a Teradata or Oracle database to the Linux or Windows server installed with Migration Tool.
2. Execute Migration Tool to migrate the syntax. In the command, specify the paths of the input file, output file, and logs.
3. Migration Tool automatically archives the migrated SQL scripts and the logs in to the specified paths.

**Figure 6-1** Migration Tool process

![Migration Tool Process Diagram]

### 6.2 Supported Keywords and Features

This section contains the system context diagram of the Migration Tool. The tool migrates source database (Teradata/Oracle) scripts to DWS. It is a command line tool operated by the migration engineer. The Migration Tool reads the scripts from the input folder. After the migration process is completed, the migrated scripts are placed in the output folder. The tool operations and errors are logged to the log folder,
6.2.1 Overview

The key to Migration Tool execution is the migration of keywords in SQL scripts. This section describes the Teradata, Oracle, and Oracle (Beta) keywords and features supported by Migration Tool, and the parameters for keyword migration. Set these parameters based on source databases, target databases, and migration scenarios.

Keywords that cannot be migrated are recorded in error logs after migration. Error logs also contain details about the scripts to which the error keywords belong, such as script names and locations. For details, see Log Reference and Troubleshooting.

6.2.2 Teradata Keywords and Features

Table 6-1 describes the Teradata keywords and features supported by Migration Tool.
The **Version** column lists the earliest version that supports a keyword or feature.

The **Remarks** column describes how to configure the migration of a keyword or feature.

<table>
<thead>
<tr>
<th>Section</th>
<th>Object and Keyword/Feature</th>
<th>Version</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema Objects</td>
<td>Table CREATE TABLE</td>
<td>V100R00 3C00</td>
<td>Configured by setting <code>session_mode</code></td>
</tr>
<tr>
<td></td>
<td>CHARACTER SET and CASESPECIFIC</td>
<td>18.1.0</td>
<td>Configured by setting <code>tdMigrateCharset-Case</code></td>
</tr>
<tr>
<td></td>
<td>VOLATILE</td>
<td>V100R00 3C00</td>
<td>Configured by setting <code>volatile</code> and <code>session_mode</code></td>
</tr>
<tr>
<td></td>
<td>SET</td>
<td>V100R00 3C00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>MULTISET</td>
<td>V100R00 3C00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>TITLE</td>
<td>V100R00 3C00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>TITLE with ALIAS</td>
<td>18.0.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>INDEX</td>
<td>V100R00 3C00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>CONSTRAINT</td>
<td>V100R00 3C00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>COLUMN STORE</td>
<td>V100R00 3C00</td>
<td>Configured by setting <code>rowstoreToColumn-store</code></td>
</tr>
<tr>
<td></td>
<td>PARTITION</td>
<td>18.0.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>ANALYZE</td>
<td>V100R00 3C30</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Data types</td>
<td>6.5.1, 18.2.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Support for less columns specified</td>
<td>18.0.0</td>
<td>-</td>
</tr>
<tr>
<td>Section</td>
<td>Object and Keyword/Feature</td>
<td>Version</td>
<td>Remarks</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------</td>
<td>---------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Index</td>
<td>V100R00 3C00</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>View</td>
<td>REPLACE VIEW</td>
<td>V100R00 3C00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>CHECK OPTION</td>
<td>V100R00 3C00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>VIEW WITH RECURSIVE</td>
<td>V100R00 3C00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>VIEW WITH ACCESS LOCK</td>
<td>V100R00 3C00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>COLLECT STATISTICS</td>
<td>V100R00 3C00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>ACCESS LOCK</td>
<td>V100R00 3C00</td>
<td>Configured by setting tdMigrateLOCKOption</td>
</tr>
<tr>
<td></td>
<td>DBC.Columns</td>
<td>18.0.0</td>
<td>Configured by custom DB scripts</td>
</tr>
<tr>
<td></td>
<td>DBC.TABLES</td>
<td>18.1.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>DBC.INDICES</td>
<td>18.1.0</td>
<td>-</td>
</tr>
<tr>
<td>DML</td>
<td>Order of clauses</td>
<td>18.0.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SELECT</td>
<td>V100R00 3C00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Order of clauses</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Extended Group By clause</td>
<td>V100R00 3C00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>TOP and SAMPLE</td>
<td>V100R00 3C00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>UPDATE</td>
<td>V100R00 3C00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>DELETE</td>
<td>V100R00 3C00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>MERGE</td>
<td>V100R00 3C00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>NAMED</td>
<td>18.0.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>ANALYZE</td>
<td>V100R00 3C30</td>
<td>-</td>
</tr>
<tr>
<td>Section</td>
<td>Object and Keyword/Feature</td>
<td>Version</td>
<td>Remarks</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------</td>
<td>---------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Query Migration Operators</td>
<td>QUALIFY</td>
<td>V100R00 3C00</td>
<td>Configured by setting <code>rowstoreToColumnstore</code></td>
</tr>
<tr>
<td></td>
<td>ALIAS</td>
<td>V100R00 3C00</td>
<td>Configured by setting <code>tdMigrateALIAS</code></td>
</tr>
<tr>
<td></td>
<td>FORMAT and CAST</td>
<td>V100R00 3C00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Short keys</td>
<td>V100R00 3C00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Object names starting with $</td>
<td>18.0.0</td>
<td>Configured by setting <code>tdMigrateDollar</code></td>
</tr>
<tr>
<td></td>
<td>IN and NOT IN Conversion</td>
<td>V100R00 3C00</td>
<td>Configured by setting <code>inToExists</code></td>
</tr>
<tr>
<td>Analytical functions</td>
<td>Analytical functions in ORDER BY</td>
<td>18.0.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Analytical functions in PARTITION BY</td>
<td>18.1.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Window functions</td>
<td>• CSUM</td>
<td>V100R00 3C00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• MDIFF</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• RANK</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ROW_NUMBER</td>
<td>-</td>
</tr>
<tr>
<td>Comparison and list operators</td>
<td>• ^= and GT</td>
<td>V100R00 3C00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>• EQ and NE</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• LE and GE</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• NOT= and LT</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• IN and NOT IN</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• IS NOT IN</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• LIKE ALL/NOT LIKE ALL</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• LIKE ANY/NOT LIKE ANY</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Section</td>
<td>Object and Keyword/Feature</td>
<td>Version</td>
<td>Remarks</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------</td>
<td>-----------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Table operators</td>
<td></td>
<td>18.0.0</td>
<td></td>
</tr>
<tr>
<td>Math Functions</td>
<td>● **</td>
<td>V100R00</td>
<td>Configured by setting tdMigrateNULLIF-ZERO and tdMigrateZEROIFNULL (18.0.0)</td>
</tr>
<tr>
<td></td>
<td>● MOD</td>
<td>3C00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● NULLIFZERO</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>● ZEROIFNULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>String Functions</td>
<td>CHAR function</td>
<td>V100R00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CHARACTERS function</td>
<td>3C00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INDEX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>STRREPLACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OREPLACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date and Time Functions</td>
<td>DATE</td>
<td>V100R00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TIMESTAMP</td>
<td>3C00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NEXT</td>
<td>18.0.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>V100R00</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3C00</td>
<td></td>
</tr>
<tr>
<td>Type Casting and Formattin g</td>
<td>CHAR</td>
<td>V100R00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>COLUMNS and COLUMNALIAS</td>
<td>3C00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Expressions</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>INT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DATE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DAY to SECOND</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DECIMAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Time interval</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Implicit type casting issues</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 6.2.3 Oracle Keywords and Features

Table 6-2 describes the Oracle keywords and features supported by Migration Tool.

- The **Version** column lists the earliest version that supports a keyword or feature.
- The **Remarks** column describes how to configure the migration of a keyword or feature.
Table 6-2 Oracle keywords and features supported by Migration Tool

<table>
<thead>
<tr>
<th>Section</th>
<th>Object and Keyword/ Feature</th>
<th>Version</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Objects</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Schema</td>
<td>Table</td>
<td>V100R00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Temporary table</td>
<td>18.1.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Global temporary tables</td>
<td>18.2.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Index</td>
<td>18.2.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>View</td>
<td>18.2.0, 6.5.1</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Sequence</td>
<td>18.2.0, 6.5.1</td>
<td>Configured by setting MigSupportSequence</td>
</tr>
<tr>
<td></td>
<td>PURGE</td>
<td>18.1.0, 18.2.0, 6.5.1</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Keyword</td>
<td>18.2.0</td>
<td>-</td>
</tr>
<tr>
<td><strong>DML</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SELECT</td>
<td></td>
<td>V100R00</td>
<td>-</td>
</tr>
<tr>
<td>INSERT</td>
<td></td>
<td>V100R00</td>
<td>-</td>
</tr>
<tr>
<td>MERGE</td>
<td></td>
<td>V100R00</td>
<td>-</td>
</tr>
<tr>
<td><strong>Pseudo</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Columns</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROWID</td>
<td></td>
<td>V100R00</td>
<td>-</td>
</tr>
<tr>
<td>ROWNUM</td>
<td></td>
<td>V100R00</td>
<td>-</td>
</tr>
<tr>
<td><strong>Outer Join</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>V100R00</td>
<td>-</td>
</tr>
<tr>
<td>OUTERT QUERY</td>
<td></td>
<td>18.2.0</td>
<td>-</td>
</tr>
<tr>
<td>CONNECT BY</td>
<td></td>
<td>18.2.0</td>
<td>-</td>
</tr>
<tr>
<td><strong>System</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Functions</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date functions</td>
<td></td>
<td>V100R00</td>
<td>Configured by custom functions in the MIGORA_EXT schema</td>
</tr>
<tr>
<td>LOB functions</td>
<td></td>
<td>V100R00</td>
<td>-</td>
</tr>
<tr>
<td>String functions</td>
<td></td>
<td>V100R00</td>
<td>-</td>
</tr>
<tr>
<td>Section</td>
<td>Object and Keyword/Feature</td>
<td>Version</td>
<td>Remarks</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------</td>
<td>-------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Analytic function s</td>
<td>RATIO_TO_REPORT</td>
<td>V100R00 3C20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Regular expression functions</td>
<td>V100R00 3C30</td>
<td>Configured by <em>custom functions in the MIG.ORA_EXT schema</em></td>
</tr>
<tr>
<td></td>
<td>Other functions</td>
<td>V100R00 3C10</td>
<td></td>
</tr>
<tr>
<td>PL/SQL</td>
<td>EDITIONABLE</td>
<td>18.2.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Variable assignment</td>
<td>V100R00 3C10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>END</td>
<td>18.1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EXCEPTION handling</td>
<td>V100R00 3C10</td>
<td>Configured by setting <em>exceptionHandler</em></td>
</tr>
<tr>
<td></td>
<td>Subtransaction handling</td>
<td>V100R00 3C10</td>
<td>Configured by setting <em>TxHandler</em></td>
</tr>
<tr>
<td></td>
<td>STRING</td>
<td>V100R00 3C10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LONG</td>
<td>V100R00 3C20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RESULT_CACHE</td>
<td>V100R00 3C10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Relational operator</td>
<td>V100R00 3C10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitution variable</td>
<td>V100R00 3C10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PARALLEL_ENABLE</td>
<td>V100R00 3C20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRUNCATE TABLE</td>
<td>V100R00 3C30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ALTER SESSION</td>
<td>V100R00 3C30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AUTONOMOUS</td>
<td>18.2.0</td>
<td></td>
</tr>
<tr>
<td>PL/SQL Collections (Using)</td>
<td>UDT TABLE</td>
<td>V100R00 3C10</td>
<td></td>
</tr>
</tbody>
</table>
## 6.2.4 Oracle (Beta) Keywords and Features

Table 6-3 describes the Oracle (Beta) keywords and features supported by Migration Tool.

- The **Version** column lists the earliest version that supports a keyword or feature.
- The **Remarks** column describes how to configure the migration of a keyword or feature.

### Table 6-3 Oracle (Beta) keywords and features supported by Migration Tool

<table>
<thead>
<tr>
<th>Section</th>
<th>Object and Keyword/Feature</th>
<th>Version</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-Defined Types)</td>
<td>UDT VArray</td>
<td>V100R00 3C10</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>UDT COUNT</td>
<td>V100R00 3C10</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>UDT RECORD</td>
<td>V100R00 3C10</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Enhancement of user-defined types</td>
<td>V100R00 3C20</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>EXTEND</td>
<td>18.2.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>RECORD</td>
<td>18.2.0</td>
<td>-</td>
</tr>
<tr>
<td>PL/SQL Packages</td>
<td>Package</td>
<td>V100R00 3C20</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Package variable</td>
<td>V100R00 3C30</td>
<td>Configured by custom functions in the MIG_ORA_EXT schema</td>
</tr>
<tr>
<td></td>
<td>PACKAGE1.FUNC1</td>
<td>18.2.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>REF CURSOR</td>
<td>V100R00 3C20</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Creating a Schema for Package</td>
<td>18.2.0</td>
<td>-</td>
</tr>
<tr>
<td>Section</td>
<td>Object and Keyword/Feature</td>
<td>Version</td>
<td>Remarks</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------</td>
<td>---------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td></td>
<td><strong>PCTINCREASE</strong></td>
<td>18.0.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>FOREIGN KEY</strong></td>
<td>18.0.0</td>
<td>Configured by setting foreignKeyHandler</td>
</tr>
<tr>
<td></td>
<td><strong>LONG data type</strong></td>
<td>18.0.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>Table whose name is same as a constraint name</strong></td>
<td>18.2.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>Temporary table</strong></td>
<td>18.0.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>Index</strong></td>
<td>18.0.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>Sequence</strong></td>
<td>18.0.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>• <strong>SEQUENCE</strong></td>
<td>18.0.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>• <strong>ALTER SEQUENCE</strong></td>
<td>18.1.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>• <strong>NEXTVAL</strong></td>
<td>18.1.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>• <strong>CURRVAL</strong></td>
<td>18.1.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>PURGE</strong></td>
<td>18.0.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>LIMIT</strong></td>
<td>18.1.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>Synonym</strong></td>
<td>18.2.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>DML</strong></td>
<td>18.1.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>INSERT</strong></td>
<td>18.1.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>• <strong>INSERT ALL</strong></td>
<td>18.1.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>• <strong>INSERT FIRST</strong></td>
<td>18.1.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>Pseudo Columns</strong></td>
<td>18.2.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>ROWNUM</strong></td>
<td>18.2.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>ROWID</strong></td>
<td>18.2.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>System Functions</strong></td>
<td>18.2.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>Date functions</strong></td>
<td>18.2.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>• <strong>TO_DATE</strong></td>
<td>18.2.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>• <strong>TIMESTAMP</strong></td>
<td>18.2.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>Other functions</strong></td>
<td>18.2.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>TABLE()</strong></td>
<td>18.2.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>SQLERRM (SQLCODE)</strong></td>
<td>18.2.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>SYSTEM PACKAGE</strong></td>
<td>18.2.0</td>
<td>-</td>
</tr>
</tbody>
</table>
6.3 System Requirements

Supported Databases

Table 6-4 lists the source databases supported by Migration Tool.

<table>
<thead>
<tr>
<th>Database</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata</td>
<td>13.1</td>
</tr>
<tr>
<td>Oracle</td>
<td>11g Release 2</td>
</tr>
<tr>
<td></td>
<td>12c Release 1</td>
</tr>
</tbody>
</table>

Software Requirements

OS requirements

Table 6-5 lists the OSs compatible with Migration Tool.

<table>
<thead>
<tr>
<th>OS</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUSE Linux Enterprise Server 11</td>
<td>SP1 (SUSE11.1)</td>
</tr>
<tr>
<td></td>
<td>SP2 (SUSE11.2)</td>
</tr>
<tr>
<td></td>
<td>SP3 (SUSE11.3)</td>
</tr>
<tr>
<td>OS</td>
<td>Version</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>SUSE Linux Enterprise Server 12</td>
<td>SP0 (SUSE12.0)</td>
</tr>
<tr>
<td></td>
<td>SP1 (SUSE12.1)</td>
</tr>
<tr>
<td></td>
<td>SP2 (SUSE12.2)</td>
</tr>
<tr>
<td></td>
<td>SP3 (SUSE12.3)</td>
</tr>
<tr>
<td>RHEL</td>
<td>6.4-x86_64 (RedHat6.4)</td>
</tr>
<tr>
<td></td>
<td>6.5-x86_64 (RedHat6.5)</td>
</tr>
<tr>
<td></td>
<td>6.6-x86_64 (RedHat6.6)</td>
</tr>
<tr>
<td></td>
<td>6.7-x86_64 (RedHat6.7)</td>
</tr>
<tr>
<td></td>
<td>6.8-x86_64 (RedHat6.8)</td>
</tr>
<tr>
<td></td>
<td>6.9-x86_64 (RedHat6.9)</td>
</tr>
<tr>
<td></td>
<td>7.0-x86_64 (RedHat7.0)</td>
</tr>
<tr>
<td></td>
<td>7.1-x86_64 (RedHat7.1)</td>
</tr>
<tr>
<td></td>
<td>7.2-x86_64 (RedHat7.2)</td>
</tr>
<tr>
<td></td>
<td>7.3-x86_64 (RedHat7.3)</td>
</tr>
<tr>
<td></td>
<td>7.4-x86_64 (RedHat7.4)</td>
</tr>
<tr>
<td>CentOS</td>
<td>6.4 (CentOS6.4)</td>
</tr>
<tr>
<td></td>
<td>6.5 (CentOS6.5)</td>
</tr>
<tr>
<td></td>
<td>6.6 (CentOS6.6)</td>
</tr>
<tr>
<td></td>
<td>6.7 (CentOS6.7)</td>
</tr>
<tr>
<td></td>
<td>6.8 (CentOS6.8)</td>
</tr>
<tr>
<td></td>
<td>6.9 (CentOS6.9)</td>
</tr>
<tr>
<td></td>
<td>7.0 (CentOS7.0)</td>
</tr>
<tr>
<td></td>
<td>7.1 (CentOS7.1)</td>
</tr>
<tr>
<td></td>
<td>7.2 (CentOS7.2)</td>
</tr>
<tr>
<td></td>
<td>7.3 (CentOS7.3)</td>
</tr>
<tr>
<td></td>
<td>7.4 (CentOS7.4)</td>
</tr>
<tr>
<td>Windows</td>
<td>7.0</td>
</tr>
</tbody>
</table>

Other software requirements
Table 6-6 lists other software requirements of Migration Tool.

### Table 6-6 Other software requirements

<table>
<thead>
<tr>
<th>Software</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDK 1.8.0_141 or later</td>
<td>Used to run Migration Tool.</td>
</tr>
<tr>
<td>Perl 5.8.8</td>
<td>Used to migrate Perl files.</td>
</tr>
<tr>
<td>Perl 5.24.3</td>
<td>Used to migrate Perl files in Windows.</td>
</tr>
</tbody>
</table>

### 6.4 Constraints and Limitations

This section describes the constraints and limitations of Migration Tool.

**General**

- Migration Tool is used only for syntax migration and not for data migration.
- If the `SELECT` clause of a subquery contains an aggregate function when the `IN` or `NOT IN` operator is converted to `EXISTS` or `NOT EXISTS`, errors may occur during script migration.

**Teradata**

- Migration Tool will migrate all occurrences of `mod` in the input scripts. If `mod` is not a keyword, it will be replaced with `%.

For example, the following input query contains `mod`:

```sql
SELECT sal
FROM employee
WHERE name LIKE 'mod (%)';
```

After the migration, the query is as follows:

```sql
SELECT sal
FROM employee
WHERE name LIKE ' % (%)';
```

- If a `case` statement containing `FORMAT` is not enclosed in parentheses, this statement will not be processed.

For example:

```sql
case when column1='0' then column1='value' end (FORMAT 'YYYYMMDD')as alias1
```

In this example, `case when column1= "0", column1= "value" end` is not enclosed in parentheses and it will not be processed.

- If `SELECT *` and `QUALIFY` clauses are both used in an input query, the migrated query returns an additional column for the `QUALIFY` clause.

For example:

```sql
SELECT * FROM dwQErrDtl_mc.C03_CORP_TIME_DPSIT_ACCT
WHERE 1 = 1
AND Data_Dt = CAST( '20150801' AS DATE FORMAT 'YYYYMMDD' )
QUALIFY ROW_NUMBER ( ) OVER ( PARTITION BY Agt_Num, Agt_Modif_Num ORDER BY NULL ) = 1;
```

Migrated query

```sql
SELECT * FROM dwQErrDtl_mc.C03_CORP_TIME_DPSIT_ACCT
WHERE 1 = 1
AND Data_Dt = CAST( '20150801' AS DATE FORMAT 'YYYYMMDD' )
QUALIFY ROW_NUMBER ( ) OVER ( PARTITION BY Agt_Num, Agt_Modif_Num ORDER BY NULL ) = 1;
```
SELECT * FROM (  
SELECT *, ROW_NUMBER( ) OVER( PARTITION BY Agt_Num, Agt_Modif_Num ORDER BY NULL ) AS ROW_NUM1  
FROM dwQErrDtl_mc.C03_CORP_TIME_DPSIT_ACCT  
WHERE 1 = 1  
AND DataDt = CAST('20150801' AS DATE)  
) Q1  
WHERE Q1.ROW_NUM1 = 1;

In the migrated query, the `ROW_NUMBER()` OVER(PARTITION BY Agt_Num and Agt_Modif_Num ORDER BY NULL) AS ROW_NUM1 column is returned additionally.

- Named references to a table in a query cannot be migrated from subqueries or functions.
  
For example, if the input query contains a table named `foo`, Migration Tool will not migrate any named references to the table from a subquery (`foo.fooid`) or when called from a function (`getfoo(foo.fooid)`).

```
SELECT * FROM foo  
WHERE foosubid IN (  
    SELECT foosubid  
    FROM getfoo(foo.fooid) z  
    WHERE z.fooid = foo.fooid  
);
```

- `ROWID` is not processed in the following case:

```
SELECT empno,ename,deptno, rowid  FROM ( SELECT * FROM employees where deptno is not null ) e , dept d WHERE d.deptno = e.deptno;
```

- The aggregate function in Window functions (RANK and ROW_NUMBER) is not available in the column list.
  
For example:

```
SELECT TIME, Region, ROW_NUMBER ( ) OVER (ORDER BY SUM(profit) DESC) AS rownumber,  
GROUPING (TIME) AS T,  
GROUPING (Region) AS R  
FROM Sales GROUP BY  
CUBE (TIME, Region)  
ORDER BY  
TIME, Region;
```

- The `INSERT FIRST` and `INSERT ALL` queries cannot be migrated if an asterisk (*) is specified in the `SELECT` statement and the `VALUES` clause is not specified.

- Migration Tool supports the `ROWNUM` condition specified at the end of the `WHERE` clause.
  
For example:

```
SELECT ROWNUM, ename,empid  
FROM employees  
WHERE empid = 10 AND deptno = 10 AND ROWNUM < 3  
ORDER BY ename;
```

- Migration Tool does not support the `ROWNUM` function in the `UPDATE` clause.
  
For example:

```
UPDATE tableName SET empno = ROWNUM  
where column = "value";
```

- Migration Tool does not support `PARTITION BY LIST` in the `INDEX` clause.
  
For example:
CREATE TABLE sales(acct_no NUMBER(5)
ORGANIZATION INDEX
INCLUDING acct_no
OVERFLOW TABLESPACE example
PARTITION BY LIST (acct_no)
(PARTITION VALUES (1)
PARTITION VALUES (DEFAULT)
TABLESPACE example);

- Migration Tool does not support the JOIN condition without a table name or table alias.
- Stored procedures and functions must end with a slash (/).
- If comment_storage_parameter is set to true, storage parameters will be commented out.
- CONNECT BY clause when appears in a query that contains multiple tables is not supported.
- For update clause when comes after rownum tool does not support.

**Oracle (Beta)**

Constraints and limitations of Migration Tool for Oracle (Beta) are as follows:

- The **INSERT FIRST** and **INSERT ALL** queries cannot be migrated if an asterisk (*) is specified in the **SELECT** statement and the **VALUES** clause is not specified.
- Migration Tool (Beta) does not support migration in the following cases and will skip unsupported statements:
  - EXECUTE IMMEDIATE statements
  - Indexed STRING datatype (String Arrays). For example:
```sql
CREATE OR REPLACE FUNCTION get_product_name_2 (prod_id NUMBER, lang_id VARCHAR2)
RETURN NVARCHAR2
IS
  /* string */ -- string type
  TYPE product_names IS TABLE OF string(50) INDEX BY PLS_INTEGER;
  vstring string(50);
  FUNCTION all_product_names (lang_id string) RETURN product_names
    RESULT_CACHE RELIES_ON (Product_Descriptions)
  IS
    all_names product_names;
  BEGIN
    vstring := 'String';
    FOR c IN (SELECT * FROM Product_Descriptions
      WHERE LANGUAGE_ID = lang_id) LOOP
      all_names(c.PRODUCT_ID) := c.TRANSLATED_NAME;
    END LOOP;
    RETURN all_names;
  END;
```

### 6.5 Installing Migration Tool

Before using Migration Tool, install it on the Linux or Windows server. Migration Tool supports **64-bit Linux**. Refer to **Table 6-5** to know more about compatible OSs.
Prerequisites

- In Linux, do not install or execute Migration Tool as a root user. To execute the `install.sh` script, you must have the permission to create folders.

- The size of the target folder must be at least four times that of the SQL files in the input folder.

  For example, if the size of the SQL files in the input folder is 100 KB, the target folder must have at least the space of 400 KB to process the SQL files.

  **NOTE**

  - To query for the available disk space of the target folder in Linux, run the following command:
    
    `df -P Folder path`

  - To query for the size of the input file in Linux, run the following command in the directory storing the file:
    
    `ls -l`

- JRE 1.8 or later and Perl have been installed. For details about the hardware and software requirements, see **System Requirements**.

To check the installed Java version and set the Java path, perform the following steps:

  a. Check whether the Java version meets requirements.
     
     `java -version`

  b. Ensure that the Java path is correctly set.

    - **Linux**
      
      1) Check whether the Java path is correctly set.
         
         `echo $JAVA_HOME`

      2) If no information is returned, add the following two lines to the `.bashrc` file of the current user, save the modification, and exit.
         
         Assume that the Java installation path is `/home/user/Java/jdk1.8.0_141`.
         
         `export JAVA_HOME=/home/user/Java/jdk1.8.0_141`
         
         `export PATH=$JAVA_HOME/bin:$PATH`

         3) Activate Java environment variables.
         
         `source ~/.bashrc`

    - **Windows**
      
      1) Right-click **My Computer** and choose **Properties** from the shortcut menu. The **System** window is displayed.

      2) Click **Advanced System Settings**. The **System Properties** dialog box is displayed.

      3) On the **Advanced** tab page, click **Environment Variables**. The **Environment Variables** dialog box is displayed.

      4) Select **Path** in the **System variables** area. Click **Edit** and check the Java installation path.

         If the Java installation path does not exist or is incorrect, append it in **Path**.

         Assume that the Java installation path is `C:\Program Files\Java\jdk1.8.0_141\bin` and the environment variable in **Path** is `c:`
Before Oracle PL/SQL objects (procedures or functions) are migrated, migrate all DDL and DML using the **Bulk** migration type. Then, migrate the scripts containing PL/SQL objects using the **BLogic** migration type.

**NOTE**

If the migration type is **Bulk**, the input file cannot contain any PL/SQL objects. Similarly, if the migration type is **BLogic**, the input files must not contain any DDL/DML.

**Procedure**

Migration Tool is an installation-free tool. You can use it after downloading and decompressing the software package.

The following procedure describes how to obtain the package. Decompress the package **MigrationTool.zip** and then **MigrationTool.rar**, you will obtain the files shown in **Table 1 Migration Tool directory**.

**In Windows:**

**Step 1** Download the software package and package verification file.
- **MigrationTool.zip**: software package

**Step 2** Decompress the **MigrationTool.zip** package.

**Step 3** Extract files from **MigrationTool.rar**.

**Step 4** Go to the **MigrationTool** directory.

**Step 5** Find and check the files in the **MigrationTool** directory.

**Table 1 Migration Tool directory** describes the obtained folders and files.

**In Linux:**

**Step 1** Decompress the **MigrationTool.zip** package.

```
unzip -ao MigrationTool.zip
```

**Step 2** Extract files from **MigrationTool.rar**.

```
sh install.sh
```

**Step 3** Go to the **MigrationTool** directory.

```
cd MigrationTool
```

**Step 4** Check the files in the **MigrationTool** directory.

```
ls config lib scripts migrate.sh migration.jar migrationtool.jar oracle-migration.jar
```

**Table 1 Migration Tool directory** describes the obtained folders and files.
Table 6-7 Migration Tool directory

<table>
<thead>
<tr>
<th>Folder or File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>config</td>
<td>Configuration files of Migration Tool</td>
</tr>
<tr>
<td>lib</td>
<td>Library files required for the normal running of Migration Tool</td>
</tr>
<tr>
<td>scripts</td>
<td>Customized configuration scripts for Oracle and Teradata migration, which can be executed to implement corresponding functions</td>
</tr>
<tr>
<td>migrationtool.jar</td>
<td>Executable files of Migration Tool in Windows</td>
</tr>
<tr>
<td>migrate.sh</td>
<td>Executable files of Migration Tool in Linux</td>
</tr>
<tr>
<td>migration.jar</td>
<td>Migration Tool core. It is invoked by executable files for Teradata and Oracle syntax migration.</td>
</tr>
<tr>
<td>oracle-migration.jar</td>
<td>Migration Tool core. It is invoked by executable files for enhanced Oracle syntax migration.</td>
</tr>
</tbody>
</table>

If you no longer use Migration Tool, you can uninstall it by deleting the following folders and files:
- config/
- lib/
- scripts/
- migration.jar
- migrate.sh
- migrationtool.jar
- oracle-migration.jar

### 6.6 Configuring the Migration Tool

#### 6.6.1 Overview

Migration Tool provides configuration files and parameters to configure migration logic and rules. Before migrating scripts, you need to configure the migration as needed. The configuration includes:

1. Tool configuration before migration: Configure Migration Tool after its installation.
2. Rule configuration before migration: Configure migration rules by modifying configuration files for Teradata SQL, Oracle SQL, Oracle (Beta) SQL, or Teradata Perl.
3. Custom configuration after migration: Execute custom DB scripts in the target database to support migration of the input keywords that do not exist in the target database.

6.6.2 Migration Tool Configuration

Configure the following items:

- **Setting application.properties**: Configure the migration behavior of Migration Tool, for example, whether to overwrite the files in the target folder and whether to format the SQL files.
- **Setting TOOL_HOME**: Configure the package path of Migration Tool.
- **Setting Java Memory Allocation**: Configure the memory that can be used by Migration Tool. If the memory usage exceeds the threshold, Migration Tool displays an error and exits.

**Setting application.properties**

The application.properties file contains the application parameters that are used to configure the behavior of Migration Tool. They are general parameters and are applicable to Teradata, Oracle, and Oracle (Beta).

Perform the following steps to configure the parameters:

**Step 1** Open the application.properties file in the config folder.

**Step 2** Set parameters in the application.properties file as needed.

**Table 6-8** describes the parameters in the application.properties file.

💻 **NOTE**

- Parameter values are case-insensitive.
- Do not modify any other parameter except the listed ones.

**Step 3** Save the configuration and exit.

----End
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
<th>Default Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>• formattedSourceRequired</td>
<td>Whether to use SQL Formatter to format the source SQL files. If this parameter is set to true, the copies of the input files are formatted and saved to the Output path formattedSource directory.</td>
<td>• true</td>
<td>true</td>
<td>formattedSourceRequired=true</td>
</tr>
<tr>
<td>• target_files</td>
<td>Operations to be performed on the output folder.</td>
<td>• overwrite</td>
<td>overwrite</td>
<td>target_files=overwrite</td>
</tr>
<tr>
<td></td>
<td><strong>Overwrite:</strong> The output folder will be overwritten.</td>
<td>• delete</td>
<td>delete</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Delete:</strong> All files in the output folder will be deleted.</td>
<td>• cancel</td>
<td>cancel</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Cancel:</strong> Cancel an operation on the output folder that has files.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------------------</td>
<td>------------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>encodingFormat</td>
<td>Encoding format of input files. If this parameter is not set (or is commented out), Migration Tool uses the default encoding format based on locale settings. <strong>NOTE</strong> The auto detection of file encoding is inaccurate. To ensure that the correct encoding format is used, specify the format using this parameter. For details about parameter settings for Oracle (Beta), see Oracle (Beta) SQL Configuration.</td>
<td>UTF8, UTF16, UTF32, ASCII and others</td>
<td>Default based on locale.</td>
<td>encodingFormat=UTF8</td>
</tr>
<tr>
<td>NoOfThreads</td>
<td>Number of threads used for migration. <strong>NOTE</strong> For details about parameter settings for Oracle (Beta), see NoOfThreads.</td>
<td>Depending on available system resources</td>
<td>3</td>
<td>NoOfThreads=3</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------</td>
<td>---------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>MaxFileSize</td>
<td>Warning threshold for the input file size (unit: B, KB, MB, or GB). If an invalid value is specified, the default value is used. The following warning will be displayed if the specified source file size exceeds the threshold: *****************[WARNING]: Migration of the following files (&gt;100KB) will take more time: bigfile001.sql bigfile008.sql *****************</td>
<td>10 KB–1 GB</td>
<td>10 MB</td>
<td>MaxFileSizeWarning=100 MB</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>MaxSqlLen</td>
<td>Maximum size of a query to be migrated.</td>
<td>If an invalid value is specified, Migration Tool resets it to the default and displays the following warning: The query length parameter (MaxSqlLen) value is out of range. Resetting to default value. If an input query exceeds the specified maximum length, the pre-validation of the query migration will fail. Migration Tool skips this query and logs the following error: 2018-07-06 12:05:57,598 ERROR TeradataBulkHandler: 195 Error occurred during processing of input in Bulk Migration. PreQueryValidation failed due to: Invalid termination; OR exclude keyword found in query; OR query exceeds maximum length (MaxSqlLen config parameter). filename.sql for Query in position : xx</td>
<td>1 to 52,428,800 bytes (1 byte to 50 MB)</td>
<td>MaxSqlLen=1048576</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------</td>
<td>-------------</td>
<td>---------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>initialJVMMemory</td>
<td>Initial memory.</td>
<td>N/A</td>
<td>256 m</td>
<td>initialJVMMem=256 m</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This indicates, the process will start up with 256 MB of memory</td>
</tr>
<tr>
<td>maxJVMMemory</td>
<td>Maximum memory.</td>
<td>N/A</td>
<td>1024 m</td>
<td>maxJVMMem=2048m</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This indicates, the process will use up to 2048 MB of memory</td>
</tr>
<tr>
<td>formatterrequired</td>
<td>If any changes in this configuration, the tool will not function as expected.</td>
<td>true, false</td>
<td>true</td>
<td>formatterrequired=true</td>
</tr>
<tr>
<td>prevalidationFlag</td>
<td>If any changes in this configuration, the tool will not function as expected.</td>
<td>true, false</td>
<td>true</td>
<td>prevalidationFlag=true</td>
</tr>
<tr>
<td>commentSeparatorFlag</td>
<td>If any changes in this configuration, the tool will not function as expected.</td>
<td>true, false</td>
<td>true</td>
<td>commentSeparatorFlag=true</td>
</tr>
<tr>
<td>queryDelimiter</td>
<td>If any changes in this configuration, the tool will not function as expected.</td>
<td>N/A</td>
<td>4 Hours</td>
<td>Timeout=4</td>
</tr>
<tr>
<td>blogicDelimiter</td>
<td>If any changes in this configuration, the tool will not function as expected.</td>
<td>N/A</td>
<td>4 Hours</td>
<td>Timeout=4</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>-------------</td>
<td>---------------</td>
<td>---------</td>
</tr>
<tr>
<td>formattedSourceRequired</td>
<td>Possible values for formatted source required can be true or false.</td>
<td>• true • false</td>
<td>true</td>
<td>formattedSourceRequired=true</td>
</tr>
<tr>
<td>target_files</td>
<td>Possible values for target files can be overwrite or delete or cancel.</td>
<td>• overwrite • delete • cancel</td>
<td>overwrite</td>
<td>target_files=overwrite</td>
</tr>
<tr>
<td>encodingFormat</td>
<td>Possible values encoding format can be UTF8 or GB2312.</td>
<td>• UTF8 • GB2312</td>
<td>UTF8</td>
<td>encodingFormat=UTF8</td>
</tr>
<tr>
<td>NoOfThreads</td>
<td>Possible values for number of threads can be any integer value based on the current system.</td>
<td>N/A</td>
<td>24</td>
<td>NoOfThreads=24</td>
</tr>
<tr>
<td>MaxFileSize Warning</td>
<td>Maximum size of the file.</td>
<td>N/A</td>
<td>10 MB</td>
<td>MaxFileSizeWarning=10 MB</td>
</tr>
<tr>
<td>MaxSqlLen</td>
<td>Maximum length of a query.</td>
<td>N/A</td>
<td>1048576</td>
<td>MaxSqlLen=1048576</td>
</tr>
<tr>
<td>initialJVMMemory</td>
<td>Initial JVM Memory.</td>
<td>N/A</td>
<td>1000 m</td>
<td>initialJVMMemory=1000 m</td>
</tr>
<tr>
<td>maxJVMMemory</td>
<td>Maximum JVM Memory.</td>
<td>N/A</td>
<td>40000 m</td>
<td>maxJVMMemory=40000 m</td>
</tr>
<tr>
<td>exceptionHandler</td>
<td>Possible values for exception commenting can be true or false.</td>
<td>• true • false</td>
<td>false</td>
<td>exceptionHandler=False</td>
</tr>
<tr>
<td>TxHandler</td>
<td>Possible values for transaction command commenting can be true or false.</td>
<td>• true • false</td>
<td>true</td>
<td>TxHandler=true</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-----------------</td>
<td>---------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>foreignKeyHandler</td>
<td>Possible values for foreign key constraint commenting can be true or false.</td>
<td>true, false</td>
<td>true</td>
<td>foreignKeyHandler=true</td>
</tr>
<tr>
<td>globalTempTable</td>
<td>Possible values for global can be global or local.</td>
<td>global, local</td>
<td>local</td>
<td>globalTempTable=LOCAL</td>
</tr>
<tr>
<td>onCommitDeleteRows</td>
<td>Possible values for delete can be DELETE or PRESERVE.</td>
<td>DELETE, PRESERVE</td>
<td>PRESERVE</td>
<td>onCommitDeleteRows=PRESERVE</td>
</tr>
<tr>
<td>maxValInSequence</td>
<td>Maximum value for sequence of the value should be greater than zero.</td>
<td>N/A</td>
<td>922337203685477580</td>
<td>maxValInSequence=922337203685477580</td>
</tr>
<tr>
<td>mergeImplementation</td>
<td>Merge Implementation has two types, one using WITH clause and the other splitting the queries. Possible values can be With or Split or None.</td>
<td>With, Split, None</td>
<td>None</td>
<td>mergeImplementation=none</td>
</tr>
<tr>
<td>RemoveListPartition</td>
<td>Possible values for LIST partition can be true or false. True value will comment the LIST partition.</td>
<td>true, false</td>
<td>true</td>
<td>RemoveListPartition=true</td>
</tr>
<tr>
<td>RemoveHashPartition</td>
<td>Possible values for HASH partition can be true or false. True value will comment the HASH partition.</td>
<td>true, false</td>
<td>true</td>
<td>RemoveHashPartition=true</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------</td>
<td>---------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>RemoveListSub Partition</td>
<td>Possible values for Subpartition of List can be true or false. True value will comment the LIST Subpartition.</td>
<td>● true • false</td>
<td>true</td>
<td>RemoveListSub Partition=true</td>
</tr>
<tr>
<td>RemoveHashSubPartition</td>
<td>Possible values for Subpartition of HASH can be true or false. True value will comment the HASH Subpartition.</td>
<td>● true • false</td>
<td>true</td>
<td>RemoveHashSubPartition=true</td>
</tr>
<tr>
<td>RemoveRange SubPartition</td>
<td>Possible values for Subpartition of RANGE can be true or false. True value will comment the RANGE Subpartition.</td>
<td>● true • false</td>
<td>true</td>
<td>RemoveRangeSubPartition=true</td>
</tr>
<tr>
<td>MigSupportSequence</td>
<td>Migration support for SEQUENCE can be true or false. True value will change create script to Insert script.</td>
<td>● true • false</td>
<td>false</td>
<td>MigSupportSequence=false</td>
</tr>
<tr>
<td>RemovePartitionTS</td>
<td>Possible values for Remove Partition TS commenting can be true or false.</td>
<td>● true • false</td>
<td>true</td>
<td>RemovePartitionTS=true</td>
</tr>
<tr>
<td>BitmapIndexSupport</td>
<td>Possible values for BitmapIndex can be comment or btree.</td>
<td>● comment • btree</td>
<td>comment</td>
<td>BitmapIndexSupport=comment</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------</td>
<td>----------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>pkgSchemaNaming</td>
<td>Possible values for pkg Schema Naming can be true or false. Schema1.package1#procedure1 should be changed to package1.procedure1. True - schema1.package1#procedure1 should be changed to package1.procedure1. False - schema1.package1#procedure1 will not be removed</td>
<td>• true, • false</td>
<td>true</td>
<td>pkgSchemaNaming=true</td>
</tr>
<tr>
<td>plsqlCollection</td>
<td>Possible values for plsqlCollection can be varray or localtable or none.</td>
<td>• varray, • localtable, • none</td>
<td>varray</td>
<td>plsqlCollection=varray</td>
</tr>
<tr>
<td>commentStorageParameter</td>
<td>Possible values for storage parameters in create table can be true or false. True value will comment storage parameters.</td>
<td>• true, • false</td>
<td>true</td>
<td>commentStorageParameter=true</td>
</tr>
<tr>
<td>MigSupportForListAgg</td>
<td>Possible values for MigSupportForListAgg can be true or false.</td>
<td>• true, • false</td>
<td>true</td>
<td>MigSupportForListAgg=true</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------------------</td>
<td>---------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>MigSupportFor RegexFunc</td>
<td>Possible values for MigSupportFor RegexFunc Replace can be true or false.</td>
<td>● true ● false</td>
<td>true</td>
<td>MigSupportFor RegexFunc=true</td>
</tr>
<tr>
<td>commentPragmaAutomationTrans</td>
<td>Possible values for commentPragmaAutomationTrans can be true or false.</td>
<td>● true ● false</td>
<td>true</td>
<td>commentPragmaAutomationTrans=true</td>
</tr>
<tr>
<td>supportJoinOperator</td>
<td>Possible values for support Join Operator can be true or false.</td>
<td>● true ● false</td>
<td>false</td>
<td>supportJoinOperator=false</td>
</tr>
<tr>
<td>migInsertWithTableAlias</td>
<td>Possible values for mig Insert With Table Alias can be true or false.</td>
<td>● true ● false</td>
<td>true</td>
<td>migInsertWithTableAlias=true</td>
</tr>
<tr>
<td>varraySize</td>
<td>Varray datatype size</td>
<td>N/A</td>
<td>1024</td>
<td>varraySize=1024</td>
</tr>
<tr>
<td>varrayObjectSize</td>
<td>Varray Object datatype size</td>
<td>N/A</td>
<td>10240</td>
<td>varrayObjectSize=10240</td>
</tr>
<tr>
<td>migrationScope</td>
<td>MigrationScope can be pkgSplit or complete Migration.</td>
<td>● pkgSplit ● completeMigration</td>
<td>completeMigration</td>
<td>migrationScope=completeMigration</td>
</tr>
<tr>
<td>migSupportConnectBy</td>
<td>Possible values for migSupportConnectBy can be true or false.</td>
<td>● true ● false</td>
<td>true</td>
<td>migSupportConnectBy=true</td>
</tr>
<tr>
<td>migSupportUnnest</td>
<td>Possible values for migSupportUnnest can be true or false.</td>
<td>● true ● false</td>
<td>true</td>
<td>migSupportUnnest=true</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------------</td>
<td>---------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>extendedGroupByClause</td>
<td>Migration support for extendedGroupByClause can be grouping sets or cube or rollup.</td>
<td>• true</td>
<td>false</td>
<td>extendedGroupByClause=false</td>
</tr>
<tr>
<td>supportDupValOnIndex</td>
<td>Migration support for DUP_VAL_ON_INDEX - UNIQUE_VIOLATION or OTHERS (older versions)</td>
<td>• UNIQUE_VIOLATION</td>
<td>UNIQUE_VIOLATION</td>
<td>supportDupValOnIndex=UNIQUE_VIOLATION</td>
</tr>
<tr>
<td>pkgvariable</td>
<td>Possible values for pkgvariable can be localtable or sys_set_context or none.</td>
<td>• localtable</td>
<td>localtable</td>
<td>pkgvariable=localtable</td>
</tr>
<tr>
<td>addPackageNameList</td>
<td>Possible values for addPackageNameList can be true or false.</td>
<td>• true</td>
<td>true</td>
<td>addPackageNameList=true</td>
</tr>
<tr>
<td>addPackageTag</td>
<td>Possible values for addPackageTag can be true or false.</td>
<td>• true</td>
<td>true</td>
<td>addPackageNameList=true</td>
</tr>
<tr>
<td>addGrantLine</td>
<td>Possible values for add Grant Line can be true or false.</td>
<td>• true</td>
<td>false</td>
<td>addGrantLine=false</td>
</tr>
<tr>
<td>MigDbmsLob</td>
<td>Possible values for MigDbmsLob can be true or false.</td>
<td>• true</td>
<td>true</td>
<td>MigDbmsLob=true</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------------------------------</td>
<td>---------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>uniqueConsForPartitionedTable</td>
<td>Unique or primary key constraint for partitioned table. Possible values can be comment_partition or comment_unique or none.</td>
<td>• comment_partition&lt;br&gt;• comment_unique&lt;br&gt;• none</td>
<td>comment_partition</td>
<td>uniqueConsForPartitionedTable=comment_partition</td>
</tr>
<tr>
<td>• volatile</td>
<td>Possible values for volatile can be local temporary or unlogged</td>
<td>• local temporary&lt;br&gt;• unlogged</td>
<td>LOCAL TEMPORARY</td>
<td>volatile=LOCAL TEMPORARY</td>
</tr>
<tr>
<td>• distributeByHash</td>
<td>Possible values for distribute by hash in output can be one or many.</td>
<td>• one&lt;br&gt;• many</td>
<td>MANY</td>
<td>distributeByHash=MANY</td>
</tr>
<tr>
<td>• rowstoreToColumnstore</td>
<td>Possible values for rowstore to columnstore can be true or false.</td>
<td>• true&lt;br&gt;• false</td>
<td>false</td>
<td>rowstoreToColumnstore=false</td>
</tr>
<tr>
<td>• inToExists</td>
<td>Query optimization for IN and NOT IN operator. Possible values can be TRUE or FALSE. TRUE value will change 'IN'/NOT IN' operator to 'EXISTS'/NOT EXISTS'</td>
<td>• true&lt;br&gt;• false</td>
<td>false</td>
<td>inToExists=false</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------------</td>
<td>---------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>session_mode</td>
<td>Applications set the session mode to achieve the most performance. Possible values can be Teradata or ANSI. In a Teradata system, a table can be either MULTISET or SET. In Teradata mode the default is SET. In ANSI mode the default is MULTISET.</td>
<td>teradata, ANSI</td>
<td>teradata</td>
<td>session_mode=teradata</td>
</tr>
<tr>
<td>deleteToTruncate</td>
<td>Query optimization for delete. Possible values can be true or false. True: Delete keyword will be changed to Truncate true.</td>
<td>true, false</td>
<td>false</td>
<td>deleteToTruncate=false</td>
</tr>
<tr>
<td>extendedGroupByClause</td>
<td>Migration support for extended group by can be grouping sets or cube or rollup.</td>
<td>grouping sets, cube, rollup</td>
<td>false</td>
<td>extendedGroupByClause=false</td>
</tr>
<tr>
<td>tdMigrateZEROIFNULL</td>
<td>Migration support for ZERO IF NULL possible values can be true or false.</td>
<td>true, false</td>
<td>true</td>
<td>tdMigrateZEROIFNULL=true</td>
</tr>
<tr>
<td>tdMigrateNULLIFZERO</td>
<td>Migration support for NULL IF ZERO possible values can be true or false.</td>
<td>true, false</td>
<td>true</td>
<td>tdMigrateNULLIFZERO=true</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------</td>
<td>---------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>• tdMigrateDollar</td>
<td>Migration support for DOLLAR symbol possible values can be true or false.</td>
<td>• true</td>
<td>true</td>
<td>tdMigrateDollar=true</td>
</tr>
<tr>
<td>• tdMigrateALIAS</td>
<td>Migration support for alias possible values can be true or false.</td>
<td>• true</td>
<td>false</td>
<td>tdMigrateALIAS=false</td>
</tr>
<tr>
<td>• tdMigrateLOCKOption</td>
<td>Migration support for Lock possible values can be true or false.</td>
<td>• true</td>
<td>false</td>
<td>tdMigrateLOCKOption=false</td>
</tr>
<tr>
<td>• tdMigrateVIEWCHECKOPTION</td>
<td>Migration support for Check possible values can be true or false.</td>
<td>• true</td>
<td>false</td>
<td>tdMigrateVIEWCHECKOPTION=false</td>
</tr>
<tr>
<td>• tdMigrateCharsetCase</td>
<td>Migration support for Chart set case possible values can be true or false.</td>
<td>• true</td>
<td>false</td>
<td>tdMigrateCharsetCase=false</td>
</tr>
<tr>
<td>• terdataUtilities</td>
<td>Migration support for Teradata utilities possible values can be true or false.</td>
<td>• true</td>
<td>true</td>
<td>terdataUtilities=true</td>
</tr>
<tr>
<td>• unique_primary_index_in_column_table</td>
<td>Creating a unique index for a column table is not supported and it can be true or false.</td>
<td>• true</td>
<td>true</td>
<td>unique_primary_index_in_column_table=true</td>
</tr>
</tbody>
</table>
### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
<th>Default Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileExtension</td>
<td>If there are any modifications in the following configuration item fileExtension then the tool will not function as expected.</td>
<td>● csv ● txt ● sql</td>
<td>sql</td>
<td>fileExtension=sql</td>
</tr>
</tbody>
</table>

#### NOTE

If a parameter is set to an incorrect or invalid value, Migration Tool uses the default value of the parameter.

### Setting TOOL_HOME

**TOOL_HOME** specifies the directory where the Migration Tool software package is decompressed. The directory must contain the `config` folder, `lib` folder, `migrate.sh` and `migration.jar` to execute Migration Tool.

#### NOTE

**TOOL_HOME** path needs to be modified if you want to execute the tool from some other path.

```bash
# If migrate.sh will be called from a different path update TOOL_HOME below
export TOOL_HOME="/home/user/test/MigrationTool"
log_path="/TOOL_HOME/log"
src="" target="" log=""
du=""
```

The following procedure describes how to set **TOOL_HOME** in Linux.

**Step 1** Run the following command to open the `migrate.sh` file:
```bash
vi migrate.sh
```

**Step 2** Press i to enter the editing mode and set **TOOL_HOME**.

`/home/user/test/MigrationTool` is used as an example.
```bash
export TOOL_HOME="/home/user/test/MigrationTool"
```

#### NOTE

If the value of **TOOL_HOME** contains spaces, enclose the value with double quotation marks (""").

For example, if the path is `/home/user/test/Migration Tool`, set **TOOL_HOME** to "'/home/user/test/Migration Tool"."
Setting Java Memory Allocation

Migration Tool has preset settings for the memory allocation of the Java Virtual Machine (JVM).

If the memory usage exceeds the limit during migration, Migration Tool displays the "java.lang.OutOfMemoryError: GC overhead limit exceeded" error and exit. In this case, you can increase the values of `initialJVMMemory` and `maxJVMMemory` in the `application.properties` file to allocate more memory.

![NOTE]

The available system resources also determine the memory allocation.

**Table 6-9 Parameters for JVM memory allocation**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Recommended Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xms</td>
<td>Initial memory allocation (unit: MB)</td>
<td>The minimum value is 256 MB. The maximum value depends on the available system resources. Default value: <strong>256</strong></td>
</tr>
<tr>
<td>Xmx</td>
<td>Upper limit for memory allocation (unit: MB)</td>
<td>The minimum value is 1024 MB. The maximum value depends on the available system resources. Default value: <strong>1024</strong></td>
</tr>
</tbody>
</table>

The following procedure describes how to set Java memory allocation in Linux.

**Step 1** Run the following command to open the `migrate.sh` file:
```
vi migrate.sh
```

**Step 2** Press i to enter the editing mode and set **Xms** and **Xmx**.

For example: Set **Xms** to **256** MB and **Xmx** to **1024** MB.
```
java -Xms256m -Xmx1024m -Dconfig.dir=$TOOL_HOME/config -Dlog4j.conf...
```

**Step 3** Press Esc, enter .wq, and press Enter to save the path and exit.

6.6.3 Teradata SQL Configuration

Teradata parameters are used to customize rules for Teradata script migration.

Open the `features-teradata.properties` file in the `config` folder and set parameters in Table 6-10 as needed.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
<th>Default Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>deleteToTruncate</td>
<td>Rule for migrating DELETE statements without a WHERE clause.</td>
<td>• true</td>
<td>false</td>
<td>deleteToTruncate=true</td>
</tr>
<tr>
<td></td>
<td>• false: Disable the migration of DELETE to TRUNCATE.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>-------------</td>
<td>---------------</td>
<td>---------</td>
</tr>
<tr>
<td>distributeByHash</td>
<td>Which columns specified in the primary index will be used for data distribution across nodes in the cluster. <strong>one</strong>: Data is distributed based on the first column specified in the primary index. <strong>many</strong>: Data is distributed based on all the columns specified in the primary index. This function is addressed by using the <code>DISTRIBUTE BY</code> clause. <strong>NOTE</strong> This parameter is set to <strong>one</strong> in V100R002C60 because this version does not support multiple columns in the <code>DISTRIBUTE BY</code> clause.</td>
<td>• one • many</td>
<td>many</td>
<td>distributeByHash =many</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------</td>
<td>---------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>• extendedGroupByClause</td>
<td>Whether to enable the migration of GROUP BY (grouping sets/cube/rollup).</td>
<td>• true</td>
<td>false</td>
<td>extendedGroupByClause=false</td>
</tr>
<tr>
<td></td>
<td><strong>true</strong>: Enable the migration of GROUP BY().</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>false</strong>: Disable the migration of GROUP BY().</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• inToExists</td>
<td>Whether to enable the migration of IN/NOT IN to EXISTS/NOT EXISTS for query optimization.</td>
<td>• true</td>
<td>false</td>
<td>inToExists=false</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• rowstoreToColumnstore</td>
<td>Whether to enable the migration of row-store tables to column-store tables.</td>
<td>• true</td>
<td>false</td>
<td>rowstoreToColumnstore=false</td>
</tr>
<tr>
<td></td>
<td>If this parameter is set to <strong>true</strong>, all row-store tables are converted to column-store tables during script migration.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------</td>
<td>---------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>session_mode</td>
<td>Default table type (SET/MULTISET) for <code>CREATE TABLE</code>.</td>
<td></td>
<td>Teradata</td>
<td>session_mode=</td>
</tr>
<tr>
<td></td>
<td><strong>Teradata</strong>: The default table type is SET.</td>
<td></td>
<td>ANSI</td>
<td>ANSI</td>
</tr>
<tr>
<td></td>
<td><strong>ANSI</strong>: The default table type is MULTISET.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tdMigrateALIAS</td>
<td>Whether to enable the migration of <code>ALIAS</code>.</td>
<td>true, false</td>
<td>false</td>
<td>tdMigrateALIAS=true</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------</td>
<td>---------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>tdMigrateDOL LAR</td>
<td>Whether to enable the migration of static objects whose name start with a dollar sign ($). This parameter is not applicable to dynamic objects, in format of ${}.</td>
<td>true, false</td>
<td>true</td>
<td>tdMigrateDOLL AR=true</td>
</tr>
</tbody>
</table>

**true**: Enclose the name of a static object with double quotation marks (").

**false**: Disable the migration of static objects.

**NOTE**
For details, see Object Names Starting With $.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
<th>Default Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>tdMigrateLOC</td>
<td>Whether to enable the migration of queries with the LOCK keyword.</td>
<td>• true</td>
<td>false</td>
<td>tdMigrateLOCK option=true</td>
</tr>
<tr>
<td></td>
<td><strong>true</strong>: Enable the migration of such queries and comment out the LOCK</td>
<td>• false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>keyword. <strong>false</strong>: Disable the migration of such queries. In this case,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Migration Tool skips this query and logs the following information: Gauss</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>does not have equivalent syntax for LOCK option in CREATE VIEW and INSERT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>statement. Please enable the config_param tdMigrateLockOption to comment</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>the LOCK syntax in the statement.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong> For details, see ACCESS LOCK.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------</td>
<td>---------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>• tdMigrateNULLIFZERO</td>
<td>Whether to enable the migration of <code>NULLIFZERO()</code>.&lt;br&gt;&lt;br&gt;<strong>true</strong>: Enable the migration of <code>NULLIFZERO()</code>.&lt;br&gt;&lt;br&gt;<strong>false</strong>: Disable the migration of <code>NULLIFZERO()</code>.&lt;br&gt;&lt;br&gt;In this case, Migration Tool copies the views and logs the following information: Gauss does not support WITH CHECK OPTION in CREATE VIEW. Please enable the config_param <code>tdMigrateViewCHECKOPTION</code> to comment the WITH CHECK OPTION syntax in the statement.</td>
<td>• true</td>
<td>• false</td>
<td>tdMigrateNULLIFZERO=true</td>
</tr>
<tr>
<td>• tdMigrateVIEWCHECKOPTI</td>
<td>Whether to enable the migration of views containing <code>CHECK OPTION</code>. &lt;br&gt;&lt;br&gt;<strong>true</strong>: Comment out such views during migration. &lt;br&gt;&lt;br&gt;<strong>false</strong>: Disable the migration of such views. In this case, Migration Tool copies the views and logs the following information: Gauss does not support WITH CHECK OPTION in CREATE VIEW. Please enable the config_param <code>tdMigrateViewCHECKOPTION</code> to comment the WITH CHECK OPTION syntax in the statement.</td>
<td>• true</td>
<td>false</td>
<td>tdMigrateVIEWCHECKOPTION=true</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------</td>
<td>---------------</td>
<td>------------------</td>
</tr>
<tr>
<td>tdMigrateZEROIFNULL</td>
<td>Whether to enable the migration of ZEROIFNULL(). true: Enable the migration of ZEROIFNULL(). false: Disable the migration of ZEROIFNULL().</td>
<td>true</td>
<td>true</td>
<td>tdMigrateZEROIFNULL=true</td>
</tr>
<tr>
<td>volatile</td>
<td>Type of tables whose data is specific to a session and is stored only for the session. When the session ends, the data and tables are deleted. The value can be local temporary or unlogged. NOTE unlogged is supported in V100R002C60 and local temporary is not.</td>
<td>local temporary</td>
<td>local temporary</td>
<td>volatile=unlogged</td>
</tr>
</tbody>
</table>

**Parameter tdMigrateZEROIFNULL**

- **true**: Enable the migration of ZEROIFNULL().
- **false**: Disable the migration of ZEROIFNULL().

**Parameter volatile**

- **local temporary**: Type of tables whose data is specific to a session and is stored only for the session. When the session ends, the data and tables are deleted. The value can be local temporary or unlogged.
- **unlogged**: NOTE unlogged is supported in V100R002C60 and local temporary is not.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
<th>Default Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>tdMigrateChar-</td>
<td>Whether to enable the migration of CHARACTER SET and CASESPECIFIC.</td>
<td>true</td>
<td>false</td>
<td>tdMigrateCharsetCase=false</td>
</tr>
<tr>
<td>setCase</td>
<td>true: Comment out CHARACTER SET and CASESPECIFIC during script migration.</td>
<td></td>
<td></td>
<td>NOTE</td>
</tr>
<tr>
<td></td>
<td>false: Disable the migration of CHARACTER SET and CASESPECIFIC.</td>
<td>false</td>
<td></td>
<td>If tdMigrateCharsetCase is set to true will comment the characterset casespecific keyword</td>
</tr>
<tr>
<td></td>
<td>In this case, Migration Tool copies CHARACTER SET and CASESPECIFIC and logs the following information with query details (such as the file name and statement position): Gauss does not have an equivalent syntax for CHARACTER SET &amp; CASE SPECIFIC option in column-level. You can rewrite this statement or set the configuration parameter tdMigrateCharsetCase to TRUE to comment the Character set &amp; Case specific</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 6.6.4 Oracle SQL Configuration

Oracle parameters are used to customize rules for Oracle script migration.

Open the `features-oracle.properties` file in the `config` folder and set parameters in **Table 6-11** as needed.

**Table 6-11 Parameters in the features-oracle.properties file**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
<th>Default Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>exceptionHandler</td>
<td>Whether to comment out exception blocks in PL/SQL.</td>
<td>true, false</td>
<td>false</td>
<td>exceptionHandler=true</td>
</tr>
<tr>
<td></td>
<td><strong>true</strong>: Comment out the exception blocks.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>false</strong>: Retain the exception blocks as they are.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong> exceptionHandler is not supported in V100R002C60.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TxHandler</td>
<td>Whether to comment out COMMIT and ROLLBACK operations in PL/SQL.</td>
<td>true, false</td>
<td>true</td>
<td>TxHandler=true</td>
</tr>
<tr>
<td></td>
<td><strong>true</strong>: Comment out the operations.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>false</strong>: Retain the operations as they are.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------</td>
<td>---------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>foreignKeyHandler</td>
<td>Whether to comment out foreign key constraints.</td>
<td>true, false</td>
<td>true</td>
<td>foreignKeyHandler=true</td>
</tr>
<tr>
<td></td>
<td><strong>true</strong>: Comment out the constraints.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>false</strong>: Retain the constraints as they are.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>globalTemporaryTable</td>
<td>Valid values are GLOBAL and LOCAL. Currently, the target database does not</td>
<td>GLOBAL, LOCAL</td>
<td>LOCAL</td>
<td>encodingFormat=LOCAL</td>
</tr>
<tr>
<td></td>
<td>support GLOBAL.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>onCommitDeleteRows</td>
<td>Valid values are DELETE and PRESERVE. Currently, the target database does</td>
<td>DELETE, PRESERVE</td>
<td>PRESERVE</td>
<td>onCommitDeleteRows=PRESERVE</td>
</tr>
<tr>
<td></td>
<td>not support DELETE.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxValInSequence</td>
<td>Maximum sequence value supported by the database. Currently, the maximum</td>
<td>1-9223372036854775807</td>
<td>9223372036854775807</td>
<td>maxValInSequence=9223372036854775807</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------</td>
<td>----------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>mergeImplemenation</td>
<td>Method for migrating a merge statement.</td>
<td>WITH, SPLIT, None</td>
<td>WITH</td>
<td>mergeImplemenation=None</td>
</tr>
<tr>
<td></td>
<td><strong>SPLIT</strong>: The merge statement is split into individual queries during migration for query optimization.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>WITH</strong>: The merge statement is migrated using a WITH clause.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RemoveHashPartition</td>
<td>Whether to comment out HASH PARTITION statements.</td>
<td>true, false</td>
<td>true</td>
<td>RemoveHashPartition=false</td>
</tr>
<tr>
<td></td>
<td><strong>true</strong>: Comment out the HASH PARTITION statements.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>false</strong>: Retain the HASH PARTITION statements as they are.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RemoveHashSubPartition</td>
<td>Whether to comment out HASH SUBPARTITION statements.</td>
<td>true, false</td>
<td>true</td>
<td>RemoveHashSubPartition=false</td>
</tr>
<tr>
<td></td>
<td><strong>true</strong>: Comment out the HASH SUBPARTITION statements.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>false</strong>: Retain HASH SUBPARTITION statements as they are.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------</td>
<td>---------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>RemoveListPartition</td>
<td>Whether to comment out LIST PARTITION statements.</td>
<td>true, false</td>
<td>true</td>
<td>RemoveListPartition=false</td>
</tr>
<tr>
<td>RemoveListSubPartition</td>
<td>Whether to comment out LIST SUBPARTITION statements.</td>
<td>true, false</td>
<td>true</td>
<td>RemoveListSubPartition=false</td>
</tr>
<tr>
<td>RemoveRangeSubPartition</td>
<td>Whether to comment out RANGESUBPARTITION statements.</td>
<td>true, false</td>
<td>true</td>
<td>RemoveRangeSubPartition=false</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------</td>
<td>---------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>MigSupportSequence</td>
<td>Whether to enable the migration of <code>SEQUENCE</code> statements.</td>
<td>● true</td>
<td>true</td>
<td>MigSupportSequence=false</td>
</tr>
<tr>
<td></td>
<td><strong>true</strong>: Enable the migration of <code>CREATE</code> to <code>INSERT</code>.</td>
<td>● false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>false</strong>: Disable the migration of <code>CREATE</code>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RemovePartitionTS</td>
<td>Whether to comment out <code>PartitionTS</code> statements.</td>
<td>● true</td>
<td>true</td>
<td>RemovePartitionTS=true</td>
</tr>
<tr>
<td></td>
<td><strong>true</strong>: Comment out the <code>PartitionTS</code> statements.</td>
<td>● false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>false</strong>: Retain the <code>PartitionTS</code> statements as they are.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BitmapIndexSupport</td>
<td>Whether to comment out for BitmapIndex <code>COMMENT</code>: will comment the entire input script</td>
<td>● comment</td>
<td>comment</td>
<td>BitmapIndexSupport=comment</td>
</tr>
<tr>
<td></td>
<td><strong>btree</strong>: will retain as they are</td>
<td>● btree</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------</td>
<td>---------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td><code>commentStorageParameter</code></td>
<td>Whether to comment out the storage parameters in a table or index.</td>
<td>true, false</td>
<td>true</td>
<td>commentStorageParameter=true</td>
</tr>
<tr>
<td></td>
<td><strong>true</strong>: Comment out the storage parameters.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>false</strong>: Retain the storage parameters as they are.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>supportJoinOperator</code></td>
<td>Whether to supportJoinOperator - TRUE/FALSE. (+) Left/Right Outer join operator support.</td>
<td>true, false</td>
<td>false</td>
<td>supportJoinOperator=false</td>
</tr>
<tr>
<td><code>MigSupportForListAgg</code></td>
<td>Whether to enable the migration of <code>ListAgg</code> statements.</td>
<td>true, false</td>
<td>true</td>
<td>MigSupportForListAgg=true</td>
</tr>
<tr>
<td></td>
<td><strong>true</strong>: Enable the migration of <code>ListAgg</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>false</strong>: Disable the migration of <code>ListAgg</code>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>MigSupportForRegexReplace</code></td>
<td>Whether to enable the migration of <code>RegexReplace</code> statements.</td>
<td>true, false</td>
<td>true</td>
<td>MigSupportForRegexReplace=true</td>
</tr>
<tr>
<td></td>
<td><strong>true</strong>: Enable the migration of <code>RegexReplace</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>false</strong>: Disable the migration of <code>RegexReplace</code>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------</td>
<td>---------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>• varrayObject Size</td>
<td>VarrayObject datatype size</td>
<td>• NA</td>
<td>10240</td>
<td>varrayObjectSize=10240</td>
</tr>
<tr>
<td>• commentPragmaAutomationTrans</td>
<td>Whether to comment out the AutomationTrans parameters in a table or index.</td>
<td>• true</td>
<td>true</td>
<td>commentPragmaAutomationTrans=true</td>
</tr>
<tr>
<td></td>
<td>true: Comment out the AutomationTrans.</td>
<td>• false</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td></td>
<td>false: Retain the AutomationTrans parameters as they are.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• insertWithValue</td>
<td>Whether to replace VALUES Keyword with SELECT</td>
<td>• true</td>
<td>true</td>
<td>insertWithValue=true</td>
</tr>
<tr>
<td></td>
<td>True value will not replace VALUES Keyword with SELECT.</td>
<td>• false</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td></td>
<td>False value will replace VALUES Keyword with SELECT.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• varraySize</td>
<td>Varray datatype size</td>
<td>NA</td>
<td>1024</td>
<td>varraySize=1024</td>
</tr>
<tr>
<td>• migrationScope</td>
<td>Whether to package split or migrate completely</td>
<td>• pkgSplit</td>
<td>complete Migration</td>
<td>migrationScope=completeMigration</td>
</tr>
</tbody>
</table>
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
<th>Default Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>migrate_ConnectBy_Unnest</td>
<td>Whether to enable the migration of <code>migrate_ConnectBy_Unnest</code> - True value will migrate connectBy and Unnest. False value will retain as they are.</td>
<td>true, false</td>
<td>true</td>
<td><code>migrate_ConnectBy_Unnest=true</code></td>
</tr>
<tr>
<td>supportDupValOnIndex</td>
<td>Whether to enable the migration support for <code>DUP_VAL_ON_INDEX</code>.</td>
<td>OTHERS(older versions), UNIQUE_VIOLATION</td>
<td></td>
<td><code>supportDupValOnIndex=UNIQUE_VIOLATION</code></td>
</tr>
<tr>
<td>pkgvariable</td>
<td>Whether to enable the migration for <code>pkgvariable</code>.</td>
<td>localtable, sys_set_context, none</td>
<td>localtable</td>
<td><code>pkgvariable = localtable</code></td>
</tr>
</tbody>
</table>

**NOTE**

Migration Tool provides parameters for deleting partitions and subpartitions because the keywords for these features are not supported currently. You can comment out the statements containing these parameters or retain them as they are during script migration.

### 6.6.5 Oracle (Beta) SQL Configuration

Oracle (Beta) automatically matches the syntax features supported by the GaussDB kernel of the corresponding version based on the version parameter in the command configured by users. In this process, you do not need to modify the configuration file.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Supported Earliest GaussDB Kernel Version</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout</td>
<td>Migration Tool configuration parameter</td>
<td>Specifies the timeout interval for Migration Tool running. The unit is hour.</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>deleteIntermediateLevels</td>
<td>Migration Tool configuration parameter</td>
<td>Specifies whether to delete the running process files during the migration of Migration Tool.</td>
<td>-</td>
<td>TRUE</td>
</tr>
<tr>
<td>cmd_prompt</td>
<td>SQL*Plus command</td>
<td>Oracle SQL Plus naming PROMPT</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>cmd_set_serveroutput</td>
<td>SQL*Plus command</td>
<td>Oracle SET SERVEROUTPUT command</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>cmd_set_define</td>
<td>SQL*Plus command</td>
<td>Oracle SET DEFINE command</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>tbl_pk_notSupersetOf_distributionKey</td>
<td>Table</td>
<td>The external primary key constraint conflicts with the Gauss distribution column.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>tbl_onCommit DeleteRows</td>
<td>Table</td>
<td>Temporary table of Oracle transactions</td>
<td>-</td>
<td>V100R007: The default value is false. V100R008 330: The default value is false.</td>
</tr>
<tr>
<td>tbl_global_temp</td>
<td>Table</td>
<td>Oracle global temporary table</td>
<td>-</td>
<td>LOCAL</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Supported Earliest GaussDB Kernel Version</td>
<td>Default Value</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>tbl_drop_purge</td>
<td>Table</td>
<td>Oracle PURGE statement DROP TABLE tblName PURGE;</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>tbl_segmentCreation</td>
<td>Table</td>
<td>The keyword that specifies the segmentCreation in the table creation statement of the Oracle database</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>tbl_storage_atr_clause</td>
<td>Table</td>
<td>Storage attributes in the table creation statements in the Oracle database</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>tbl_supplementalLogData</td>
<td>Table</td>
<td>Oracle table-level supplemental logs</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>tbl_idxOrganized</td>
<td>Table</td>
<td>Oracle index organization table</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>tbl_physical_attr_clause</td>
<td>Table</td>
<td>Physical attributes in the table creation statement in the Oracle database</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>sysfn_regexp_instr</td>
<td>System function</td>
<td>Oracle REGEXP_INSTR system function</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>sysfn_regexp_replace</td>
<td>System function</td>
<td>Oracle REGEXP_REPLACE system function</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Supported Earliest GaussDB Kernel Version</td>
<td>Default Value</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------</td>
<td>------------------------------------</td>
<td>------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>sysfn_regexp_substr</td>
<td>System function</td>
<td>Oracle REGEXP_SUBSTR system function</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>sysfn_wm_concat</td>
<td>System function</td>
<td>Oracle WMSYS.WM_CONCAT system function</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>sysfn_add_months</td>
<td>System function</td>
<td>Oracle ADD_MONTHS system function</td>
<td>V100R007</td>
<td>V100R007: The default value is false. V100R008 330: The default value is false.</td>
</tr>
<tr>
<td>sysfn_months_between</td>
<td>System function</td>
<td>Oracle MONTHS_BETWEEN system function</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>sysfn_last_day</td>
<td>System function</td>
<td>Oracle LAST_DAY system function</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>sysfn_ratio_to_report</td>
<td>System function</td>
<td>Oracle RATIO_TO_REPORT() OVER() system (analysis) function</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>sysfn_userenv</td>
<td>System function</td>
<td>Oracle USERENV system function</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>sysfn_dbms_lob_createtemporary</td>
<td>System function</td>
<td>Oracle DBMS_LOB.CREATETEMPORARY system function</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>sysfn_dbms_lob_freetemporary</td>
<td>System function</td>
<td>Oracle DBMS_LOB.FREETEMPORARY system function</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Supported Earliest GaussDB Kernel Version</td>
<td>Default Value</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>sysfn_dbms_lob_append</td>
<td>System function</td>
<td>Oracle DBMS_LOB.APP END system function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sysfn_dbms_lob_instr</td>
<td>System function</td>
<td>Oracle DBMS_LOB.INS TR system function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sysfn_table</td>
<td>System function</td>
<td>Oracle TABLE system function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sysfn_systimestamp</td>
<td>System function</td>
<td>Oracle SYSTIMESTAMP system function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sysfn_sysdate</td>
<td>System function</td>
<td>Oracle SYSDATE system function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sysfn_sys_guid</td>
<td>System function</td>
<td>Oracle SYS_GUID system function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sysfn_stragg</td>
<td>System function</td>
<td>Oracle SYS.STRAGG system function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sysfn_to_clob</td>
<td>System function</td>
<td>Oracle TO_CLOB system function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>qry_tbl_alias_w_brackets_in_from_clause</td>
<td>Query statement</td>
<td>In the following statement, the FROM statement uses brackets to restrict the table name and table alias. SELECT e.ename,d.dept no FROM (emp e) , (dept) d;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Supported Earliest GaussDB Kernel Version</td>
<td>Default Value</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>exec_purge_recyclebin</td>
<td>DDL statement</td>
<td>Oracle PURGE statements PURGE RECYCLEBIN; PURGE DBA_RECYCLEBIN; PURGE TABLESPACE tsName USER userName;</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>exec_alterSessionParallel</td>
<td>DDL statement</td>
<td>Enable sessions in parallel in Oracle PL/SQL.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>qry_order_for_having_groupBy_clause</td>
<td>DML statement</td>
<td>During aggregation queries, the Oracle database does not distinguish the sequence of the GROUP BY and HAVING clauses.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>insert_first</td>
<td>DML statement</td>
<td>Oracle INSERT FIRST statement</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>qry_merge</td>
<td>DML statement</td>
<td>Oracle MERGE statement</td>
<td>V100R008 330</td>
<td>V100R007: The default value is true. V100R008 330: The default value is false.</td>
</tr>
<tr>
<td>other_implicit_conversion</td>
<td>DML statement</td>
<td>Scenarios where the static constant value is used in the Oracle SELECT statement</td>
<td>V100R008 330</td>
<td>V100R007: The default value is true. V100R008 330: The default value is false.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Supported Earliest GaussDB Kernel Version</td>
<td>Default Value</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>qry_unique</td>
<td>DML statement</td>
<td>UNIQUE keyword in Oracle SELECT (similar to DISTINCT)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>isrt_w_tbl_alias</td>
<td>DML statement</td>
<td>Table alias in the Oracle INSERT statement</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>isrt_all</td>
<td>DML statement</td>
<td>Oracle INSERT ALL statement</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>pkg_naming</td>
<td>Package</td>
<td>Indicates the packet splitting (Oracle Package) solution for Migration Tool. The value can be REMOVE_SCHEMA, MERGE_PACKAGE, or false.</td>
<td>-</td>
<td>MERGE_PACKAGE</td>
</tr>
<tr>
<td>pkg_editionable</td>
<td>Package</td>
<td>When the Oracle creates a package, the EDITIONABLE (an editable object that is related to the version feature) keyword is used to define the package attribute.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>opr_outerJoin</td>
<td>Operator</td>
<td>Oracle uses the plus sign (+) as the external connection identifier in the WHERE clause.</td>
<td>V100R008 330</td>
<td>V100R007: The default value is true. V100R008 330: The default value is false.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Supported GaussDB Kernel Version</td>
<td>Default Value</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>opr_forwardSlash_at_vw_end</td>
<td>Operator</td>
<td>When creating the VIEW, the Oracle database uses slashes (/) as the terminator (generally, the semicolon is used as the statement terminator). However, Gauss does not support this feature.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>opr_qLiteral</td>
<td>Operator</td>
<td>The Oracle SELECT statement uses the q to declare the static constant.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>sp_refcur_openFor_with_clause</td>
<td>Procedure</td>
<td>The SQL Query statement containing the CTE structure appears in the cursor scenario.</td>
<td>V100R008 330</td>
<td>V100R007: The default value is true. V100R008 330: The default value is false.</td>
</tr>
<tr>
<td>table_userDefinedDatatypes</td>
<td>Collection</td>
<td>Oracle users define the object (OBJECT) type: CREATE TYPE tyName AS OBJECT().</td>
<td>V100R007</td>
<td>V100R007: The default value is false. V100R008 330: The default value is false.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Supported Earliest GaussDB Kernel Version</td>
<td>Default Value</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>plsqlCollection</td>
<td>Collection</td>
<td>Specifies the switch of the Oracle collection migration. Parameters <strong>varraySize</strong> and <strong>varrayObjectSize</strong> are involved.</td>
<td>-</td>
<td>TRUE</td>
</tr>
<tr>
<td>varraySize</td>
<td>Collection</td>
<td>Oracle collection: Specifies the migration size of the collection composed of Associative Array, Nested Table, or VARRAY that consists of basic data types such as INTEGER and VARCHAR2. <strong>plsqlCollection</strong> indicates whether to enable this feature.</td>
<td>-</td>
<td>1024</td>
</tr>
<tr>
<td>varrayObjectSize</td>
<td>Collection</td>
<td>Oracle collection: Specifies the migration size of the collection of Nested Table that consists of RECORD or OBJECT. <strong>plsqlCollection</strong> indicates whether to enable this feature.</td>
<td>-</td>
<td>10240</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Supported Earliest GaussDB Kernel Version</td>
<td>Default Value</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>trans_rollback</td>
<td>Transaction</td>
<td>Oracle PL/SQL transaction control</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>trans_commit</td>
<td>Transaction</td>
<td>Oracle PL/SQL transaction control</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>vw_force</td>
<td>View</td>
<td>The FORCE keyword is used to create a view in the Oracle database.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>typ_string</td>
<td>Data type</td>
<td>Oracle PL/SQL STRING data type</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>typ_long</td>
<td>Data type</td>
<td>Oracle LONG data type</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>typ_pls_integer</td>
<td>Data type</td>
<td>Oracle PL/SQL PLS_INTEGER data type</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>typ_timestamp</td>
<td>Data type</td>
<td>Oracle TIMESTAMP data type</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>typ_char_w_n_char</td>
<td>Data type</td>
<td>Oracle CHAR(n CHAR) data type</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>typ_float</td>
<td>Data type</td>
<td>Oracle FLOAT data type</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>typ_longRaw</td>
<td>Data type</td>
<td>Oracle LONG RAW data type</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>typ_number_a sterisk</td>
<td>Data type</td>
<td>Oracle NUMBER(*,0) data type</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Supported Earliest GaussDB Kernel Version</td>
<td>Default Value</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>idx_create_w_schema</td>
<td>Index</td>
<td>When an index is created, SCHEMA, that is, SchemaName.IndexName, is added before the index name. Specifying a schema before the index name is not allowed in Gauss.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>idx_bmp</td>
<td>Index</td>
<td>Oracle bitmap index</td>
<td>-</td>
<td>comment</td>
</tr>
<tr>
<td>syn_synonym</td>
<td>Synonym</td>
<td>Synonym creation in the Oracle database</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>rowid</td>
<td>Pseudo column</td>
<td>Oracle ROWID pseudo-column</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>rownum</td>
<td>Pseudo column</td>
<td>Oracle ROWNUM pseudo-column</td>
<td>V100R007</td>
<td>V100R007: The default value is true. V100R008 330: The default value is false.</td>
</tr>
<tr>
<td>syspkg_utl_file</td>
<td>System package</td>
<td>Oracle SYS.UTL_FILE system package</td>
<td>V100R008 330</td>
<td>V100R007: The default value is true. V100R008 330: The default value is false.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Supported Earliest GaussDB Kernel Version</td>
<td>Default Value</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>seq_create</td>
<td>Sequence</td>
<td>The CREATE keyword is used when the Oracle database creates a sequence.</td>
<td>V100R007</td>
<td>V100R007: The default value is false. V100R008 330: The default value is false.</td>
</tr>
<tr>
<td>seq_nextval</td>
<td>Sequence</td>
<td>The Oracle uses the NEXTVAL keyword to use the sequence.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>seq_maxvalue</td>
<td>Sequence</td>
<td>Default maximum value of the sequence</td>
<td>-</td>
<td>9223372036854770000</td>
</tr>
<tr>
<td>expc_sqlerrm_w_sqlcode</td>
<td>Exception</td>
<td>Oracle exception handling function. SQLCODE is used to obtain the error number, and SQLERRM is used to obtain the error information.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>expc_dup_val_on_index</td>
<td>Exception</td>
<td>Oracle pre-definition exception. The DUP_VAL_ON_INDEX violates the primary key or unique index.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Supported Earliest GaussDB Kernel Version</td>
<td>Default Value</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>excp_invalid_number</td>
<td>Exception</td>
<td>Oracle pre-definition exception. The value of INVALID_NUMBER is abnormal. In an SQL statement, the string does not represent a valid number. As a result, an error occurs when the string is converted to a digit. (In the process statement, the error VALUE_ERROR is thrown.) This exception is thrown when the value behind the LIMIT clause of the FETCH statement is not a positive number.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>excp_zero_divide</td>
<td>Exception</td>
<td>Oracle pre-definition exception. An exception occurs when ZERO_DIVIDE is divided by zero.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Supported GaussDB Kernel Version</td>
<td>Default Value</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
<td>---------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>fn_end_stmt_w_fn_name</td>
<td>User-defined function</td>
<td>#True will handle [function</td>
<td>procedure</td>
<td>package] end statement with [function</td>
</tr>
<tr>
<td>fn_result_cache</td>
<td>User-defined function</td>
<td>In custom functions, the dataset cache RESULT_CACHE can reduce the number of internal query invoking times and improve the running efficiency.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>fn_parallelenable</td>
<td>User-defined function</td>
<td>The parallel PARALLEL_ENABLE function is enabled in the custom function of the Oracle database.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Supported Earliest GaussDB Kernel Version</td>
<td>Default Value</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>cons_name_sameAs_tbl_name</td>
<td>Constraint</td>
<td>In the Oracle table creation statement, the constraint name is the same as the table name (including the external primary key constraint, inline primary key constraint, and inline unique constraint).</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>cons_modify_col_not_null</td>
<td>Constraint</td>
<td>Setting a non-null constraint for a column in the Oracle database</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>cons_inline_usingIndex</td>
<td>Constraint</td>
<td>USING INDEX is used in the Oracle inline constraint to specify the associative index.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>cons_fk</td>
<td>Constraint</td>
<td>Oracle foreign key constraint</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>cons_enable</td>
<td>Constraint</td>
<td>The ENABLE keyword is used in the Oracle constraint.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Supported Earliest GaussDB Kernel Version</td>
<td>Default Value</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>sp_call_wo_param</td>
<td>Procedure</td>
<td>If the Oracle database invokes other stored procedures without parameters in a stored procedure, the parentheses can be omitted.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>sp_autonomous_trans</td>
<td>Procedure</td>
<td>Oracle autonomous transaction</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ptbl_list</td>
<td>Partition table</td>
<td>Oracle LIST partition table</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ptbl_range_w_interval</td>
<td>Partition table</td>
<td>Range partition table of the Oracle INTERVAL type</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ptbl_hash</td>
<td>Partition table</td>
<td>Oracle hash partition table</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ptbl_hash_w_sub</td>
<td>Partition table</td>
<td>Oracle hash sub-partition table</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ptbl_list_w_sub</td>
<td>Partition table</td>
<td>Oracle list sub-partition table</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ptbl_range_w_sub</td>
<td>Partition table</td>
<td>Oracle range sub-partition table</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ptbl_physical_attr_clause</td>
<td>Partition table</td>
<td>Physical attributes in the Oracle partition table</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Supported Earliest GaussDB Kernel Version</td>
<td>Default Value</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>addAsToKeyword</td>
<td>Keyword</td>
<td>The keyword involved in this feature is the Gauss reserved word. If the keyword is used as the column alias, the &quot;AS&quot; must be explicitly indicated. The following three scenarios are involved: 1) Column alias in the common DML SELECT statement 2) Column alias in the PL/SQL SELECT INTO statement 3) Column alias in the SELECT clause of a DDL statement for creating a view or creating a table addAsToKeyword indicates whether to enable this function. migrateKeywordUsingAS indicates the list of keywords involved in this feature.</td>
<td>-</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

addAsToKeyword indicates whether to enable this function.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Supported Earliest GaussDB Kernel Version</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>migrateKeywordUsingAS</td>
<td>Keyword configuration parameter</td>
<td>The keyword involved in this feature is the Gauss reserved word. If the keyword is used as the column alias, the &quot;AS&quot; must be explicitly indicated. The following three scenarios are involved: 1) Column alias in the common DML SELECT statement 2) Column alias in the PL/SQL SELECT INTO statement 3) Column alias in the SELECT clause of a DDL statement for creating a view or creating a table migrateKeywordUsingAS indicates the list of keywords involved, and addAsToKeyword indicates the switch of this feature.</td>
<td>-</td>
<td>• owner</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• key</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• begin</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• label</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• version</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• text</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• year</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• position</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• function</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• end</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• procedure</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• limit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• split</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• app</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• action</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• role</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• comments</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• fileheader</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• language</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• header</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• content</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• show</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• maxvalue</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• offset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• return</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• time</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• authorizati on</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• range</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• account</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Supported Earliest GaussDB Kernel Version</td>
<td>Default Value</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>addDoubleQuoteToKeyword</td>
<td>Keyword</td>
<td>If the Oracle database object is the Gauss reserved word, Migration Tool adds double quotation marks to it during migration. This feature involves object names in DDL statements created for various databases, including table names, column names, view names, function names, package names, process names, and index names. addDoubleQuoteToKeyword indicates whether to enable this function. migrateKeywordUsingQuote indicates the list of keywords involved in this feature.</td>
<td>-</td>
<td>TRUE</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Supported Earliest GaussDB Kernel Version</td>
<td>Default Value</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>migrateKeywordUsingQuote</td>
<td>Keyword configuration parameter</td>
<td>If the Oracle database object is the Gauss reserved word, Migration Tool adds double quotation marks to it during migration. This feature involves object names in DDL statements created for various databases, including table names, column names, view names, function names, package names, process names, and index names. migrateKeywordUsingQuote indicates the list of keywords involved, and addDoubleQuotationToKeyword indicates the switch of this feature.</td>
<td>-</td>
<td>● function</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>● end</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>● procedure</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>● limit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>● split</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>● maxvalue</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>● offset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>● return</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>● type</td>
</tr>
<tr>
<td>alter_idx_with_logging_clau se</td>
<td>Index</td>
<td>Oracle index log mode. Migration Tool supports the LOGGING and NOLOGGING parameters.</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
### 6.6.6 Teradata Perl Configuration

Teradata Perl parameters are used to customize rules for Teradata Perl script migration.

Open the `perl-migration.properties` file in the `config` folder and set parameters in Table 6-13 as needed.

- Parameter values are case-insensitive.
- You are not allowed to change any parameter values other than the ones in the following table.
Table 6-13 Parameters in the perl-migration.properties file

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
<th>Default Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>• add-timing-on</td>
<td>Whether to enable the insertion of scripts to calculate execution time. If it is enabled, the script will be added to each input file.</td>
<td>• true</td>
<td>false</td>
<td>add-timing-on=true</td>
</tr>
</tbody>
</table>
| • db-script-tag-name | Scripts to be processed in Perl files.  
  **BTEQ:** Only the scripts under the BTEQ tag will be processed.  
  **SQL_LANG:** Only the scripts under the SQL_LANG tag will be processed.  
  **BTEQ, SQL_LANG:** The scripts under the BTEQ and SQL_LANG tags will be processed. | • bteq       | bteq,sql_lang | db-script-tag-name=sql_lang |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
<th>Default Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>log-file-count</td>
<td>Maximum number of log files retained, including the log files in use and archived log files. If the number of log files exceeds the upper limit, the earliest files will be deleted until the new log files are successfully archived.</td>
<td>3 - 10</td>
<td>5</td>
<td>log-file-count=10</td>
</tr>
<tr>
<td>log-file-size</td>
<td>Maximum file size. Upon reaching the specified size, a file is archived by adding a timestamp to the file name. Example: perlmigrationtool_2018-07-08_16_12_08.log After the archiving, a new log file perlmigrationtool.log with a timestamp is generated.</td>
<td>1 MB - 10 MB</td>
<td>5 MB</td>
<td>log-file-size=10 MB</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------</td>
<td>---------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>logging-level</td>
<td>Logging level of Teradata Perl migration log files.</td>
<td>error</td>
<td>info</td>
<td>logging-level=info</td>
</tr>
<tr>
<td></td>
<td>error: Log only errors.</td>
<td>warning</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>warning: Log errors and warnings.</td>
<td>info</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>info: Log errors, warnings, and activity information. This level contains all log information.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------</td>
<td>---------------</td>
<td>----------------------------------</td>
</tr>
</tbody>
</table>
| migrate-variables | Whether to enable the migration of Perl variables containing SQL statements.Perl files can contain Perl variables with SQL statements. These variables are executed by using the PREPARE and EXECUTE statement in Perl. Migration Tool can extract SQL statements from Perl variables and migrate them.  
**true**: Enable the migration of Perl variables containing SQL statements.  
**false**: Disable the migration of Perl variables containing SQL statements. | ● true  
● false      | true                       | migrate-variables=true |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
<th>Default Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>databasename = &quot;$ (AUTO_DQDB)&quot; and tablename like 'V_%' order by 1;</td>
<td>$sth_rundq = $dbh-&gt;execute_query($onesql);</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Output</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$onesql = &quot;SELECT TRIM( tablename ) FROM dbc.tables WHERE databasename = &quot;$ (AUTO_DQDB)&quot; AND tablename LIKE 'V_%' ORDER BY 1;&quot;;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$sth_rundq = $dbh-&gt;execute_query($onesql);</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>● remove-intermediate-files</td>
<td>Whether to delete the intermediate SQL file generated by the Perl Migration Tool after the migration is complete. The intermediate files contain the BTEQ and SQL_LANG syntax in SQL files. These files are used as input for Migration Tool. <strong>true</strong>: Delete the intermediate files. <strong>false</strong>: Do not delete the intermediate files.</td>
<td>true/false</td>
<td>true</td>
<td>remove-intermediate-files=true</td>
</tr>
</tbody>
</table>
### 6.6.7 Custom DB Script Configuration

Custom scripts are SQL files used to migrate from Teradata/Oracle the input keywords that do not exist in the target database.

These scripts must be executed in each target database before the migration.

**Custom DB scripts**

Open the `scripts` folder in the release package. Table 6-14 lists the folders and files in the `scripts` folder.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
<th>Default Value</th>
<th>Example</th>
</tr>
</thead>
</table>
| `migrate-executequery` | Whether to enable the migration of `execute_query` containing SQL statements.  
  **true**: Enable the migration of `execute_query` containing SQL statements.  
  **false**: Disable the migration of `execute_query` containing SQL statements.  
  For example:  
  `migrate-executequery` is set to **true** and input is as follows:  
  my $rows1=$conn1->execute_query("select $ {selectclause} from $ {databasename}.${tablename};");  
  **Output**  
  my $rows1=$conn1->execute_query("SELECT ${selectclause} FROM $ (databasename).${tablename} ;"); | • true  
  • false | true | `migrate-executequery =true` |
The SQL files contain the scripts for the custom migration functions. These functions are required in DWS to support specific features of Teradata and Oracle.

Table 6-14 Custom DB scripts for Migration Tool

<table>
<thead>
<tr>
<th>Folder</th>
<th>Script File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-- scripts</td>
<td>-</td>
<td>Folder: all scripts</td>
</tr>
<tr>
<td>------- oracle</td>
<td>-</td>
<td>Folder: Oracle functions and scripts</td>
</tr>
<tr>
<td>---------- sequence</td>
<td>-</td>
<td>Folder: scripts to configure Oracle sequences</td>
</tr>
<tr>
<td>-</td>
<td>sequence_scripts.sql</td>
<td>Script: used to enable migration of Oracle sequence</td>
</tr>
<tr>
<td>------- package</td>
<td>-</td>
<td>Folder: scripts to configure Oracle package variables</td>
</tr>
<tr>
<td>-</td>
<td>pkg_variable_scripts.sql</td>
<td>Script: used to enable migration of Oracle package variables</td>
</tr>
<tr>
<td>------- function</td>
<td>-</td>
<td>Folder: scripts to configure Oracle system functions</td>
</tr>
<tr>
<td>-</td>
<td>date_functions.sql</td>
<td>Script: used to enable migration of Oracle date functions</td>
</tr>
<tr>
<td>-</td>
<td>environment_functions.sql</td>
<td>Script: used to enable migration of Oracle environment functions</td>
</tr>
<tr>
<td>-</td>
<td>string_functions.sql</td>
<td>Script: used to enable migration of Oracle string functions</td>
</tr>
<tr>
<td>------- teradata</td>
<td>-</td>
<td>Folder: Teradata functions and scripts</td>
</tr>
<tr>
<td>------- view</td>
<td>-</td>
<td>Folder: scripts to configure views</td>
</tr>
<tr>
<td>-</td>
<td>vw_td_dbc_tables.sql</td>
<td>Script: used to enable migration of Teradata DBC.TABLES</td>
</tr>
<tr>
<td>-</td>
<td>vw_td_dbc_indices.sql</td>
<td>Script: used to enable migration of Teradata DBC.INDICES</td>
</tr>
</tbody>
</table>
## Executing Custom DB Scripts

**Step 1** Use either of the following methods to execute the required scripts in all target databases for which migration is to be performed:

- **Use gsql.**
  - Use `gsql` to connect to the target database and paste all contents in the SQL file to `gsql`, which will automatically execute the pasted contents.
  
  Run the following command to connect to the database:
  ```
gsql -h <host_addr_xxx.xxx.xxx.xxx> -d <database_name> -U <user_name> -W <password> -p <port_number> -r
```

- Use `gsql` to connect to the target database and execute a SQL file.
  
  Run the following command to connect to the database and run the SQL file:
  ```
gsql -h <host_addr_xxx.xxx.xxx.xxx> -d <database_name> -U <user_name> -W <password> -p <port_number> -f <filename.sql> -o <output_filename> -L <log_filename.log> -r
```

- **Use Data Studio.**
  
  Use Data Studio to connect to the target database, and then open and run the SQL file in Data Studio.

----End
6.7 Migration Process

6.7.1 Overview

This section contains the following topics that provide an overview of the migration process and detailed steps for using the Migration Tool.

The following use cases for migration are supported by the migration tool:

- Migrate Teradata SQL
- Migrate Oracle SQL
- Migrate Oracle SQL using the new tool (Oracle Beta version)
- Migrate Teradata Perl files

**Figure 6-3** Overview of the Syntax Migration Process

This section contains information about the prerequisites to be completed before starting the migration process.

**Execute Custom Scripts**

The Migration Tool configuration contains the following custom DB scripts in the `MigrationTool/scripts`:

- `date_functions.sql`: Custom DB script for Oracle date functions.
- `environment_functions.sql`: Custom DB script for Oracle environment functions.
- `string_functions.sql`: Custom DB script for Oracle string functions.
- `pkg_variable_scripts.sql`: Custom DB script for Oracle package variable functions.
- `sequence_scripts.sql`: Custom DB script for Oracle sequence functions.
- `mig_fn_get_datatype_short_name.sql`: Custom DB script for Teradata functions.
- `mig_fn_castasint.sql`: Custom DB script for migration of CAST AS INTEGER.
- `vw_td_dbc_tables.sql`: Custom DB script for migration of DBC.TABLES.
• **vw_td_dbc_indices.sql**: Custom DB script for migration of DBC.INDICES.

These DB scripts are required to support certain input keywords not present in one or more versions of the target DB. These scripts need to be executed once in the target DB prior to migration.

For more information on executing the custom DB scripts, see [Custom DB Script Configuration](#).

Use either of the following methods to execute the required scripts in all target DWS databases for which migration is to be performed:

• Use `gsql` to connect to the DWS database and paste all contents in the `.sql` file to `gsql`, which will automatically execute the pasted content.

  ```
  gsql -h <host_addr_xxx.xxx.xxx.xxx> -d <database_name> -U <user_name> -W <password> -p <port_number> -r
  ```

• Use `gsql` to connect to the DWS database and execute a `.sql` file.

  ```
  gsql -h <host_addr_xxx.xxx.xxx.xxx> -d <database_name> -U <user_name> -W <password> -p <port_number> -f <filename.sql> -o <output_filename> -L <log_filename.log> -r
  ```

• Use Data Studio to connect to the DWS database, then open and execute the `.sql` file in Data Studio.

### Configure Migration Tool and Migration Properties

The Migration Tool configuration contains the following configuration files in the `MigrationTool/config` folder:

- **application.properties**: Configuration parameters for the Migration Tool.
- **features-teradata.properties**: Configuration parameters for Teradata SQL Migration.
- **features-oracle.properties**: Configuration parameters for Oracle SQL Migration.
- **oracle-migration.properties**: Configuration parameters for Oracle (Beta) SQL Migration.
- **perl-migration.properties**: Configuration parameters for the Perl Migration.

For more information on updating the configuration parameters, see [Migration Tool Configuration](#).

### 6.7.2 Prerequisites

#### Executing Custom DB Scripts

Custom scripts are executed to support input keywords that do not exist in certain versions of the target database. These scripts must be executed in each target database before the migration.

**Table 6-15** describes the custom scripts in the `MigrationTool/scripts` directory. For details about how to execute custom scripts, see [Custom DB Script Configuration](#).
Table 6-15 Custom DB scripts

<table>
<thead>
<tr>
<th>Script</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date_functions.sql</td>
<td>Custom DB script for Oracle date functions</td>
</tr>
<tr>
<td>environment_functions.sql</td>
<td>Custom DB script for Oracle environment functions</td>
</tr>
<tr>
<td>string_functions.sql</td>
<td>Custom DB script for Oracle string functions.</td>
</tr>
<tr>
<td>pkg_variable_scripts.sql</td>
<td>Custom DB script for Oracle package variable functions</td>
</tr>
<tr>
<td>sequence_scripts.sql</td>
<td>Custom DB script for Oracle sequence functions</td>
</tr>
<tr>
<td>mig_fn_get_datatype_short_name.sql</td>
<td>Custom DB script for Teradata functions</td>
</tr>
<tr>
<td>mig_fn_castasint.sql</td>
<td>Custom DB script for migration of CAST AS INTEGER</td>
</tr>
<tr>
<td>vw_td_dbc_tables.sql</td>
<td>Custom DB script for migration of DBC.TABLES</td>
</tr>
<tr>
<td>vw_td_dbc_indices.sql</td>
<td>Custom DB script for migration of DBC.INDICES</td>
</tr>
</tbody>
</table>

Configuring Migration Tool and Migration Properties

To configure Migration Tool, configure parameters in the configuration files in the `config` folder of MigrationTool. Table 6-16 describes the parameters.

Table 6-16 Parameters for configuring Migration Tool

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Configuration File</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Teradata SQL Migration</strong></td>
<td><strong>Migration Tool</strong>: application.properties</td>
<td>deleteToTruncate=True/False</td>
</tr>
<tr>
<td></td>
<td><strong>Teradata</strong>: features-teradata.properties</td>
<td>distributeByHash=one/many</td>
</tr>
<tr>
<td></td>
<td></td>
<td>extendedGroupByClause=True/False</td>
</tr>
<tr>
<td></td>
<td></td>
<td>inToExists=True/False</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rowstoreToColumnstore=True/False</td>
</tr>
<tr>
<td></td>
<td></td>
<td>session_mode=Teradata/ANSI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tdMigrateDollar=True/False</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tdMigrateALIAS=True/False</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tdMigrateNULLIFZero=True/False</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tdMigrateZEROIFNULL=True/False</td>
</tr>
<tr>
<td></td>
<td></td>
<td>volatile=local temporary/unlogged</td>
</tr>
</tbody>
</table>
### Scenario

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Configuration File</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Oracle SQL Migration</strong></td>
<td>• Migration Tool: application.properties&lt;br&gt;• Oracle SQL Configuration: eatures-oracle.properties</td>
<td>exceptionHandler=True/False&lt;br&gt;TxHandler=True/False&lt;br&gt;foreignKeyHandler=True/False&lt;br&gt;globalTempTable=GLOBAL/LOCAL&lt;br&gt;onCommitDeleteRows=Delete/Preserve&lt;br&gt;maxValInSequence=0..9223372036854775807&lt;br&gt;mergeImplementation=WITH/SPLIT&lt;br&gt;RemoveHashPartition=True/False&lt;br&gt;RemoveHashSubPartition=True/False&lt;br&gt;RemoveListPartition=True/False&lt;br&gt;RemoveListSubPartition=True/False&lt;br&gt;RemoveRangeSubPartition=True/False&lt;br&gt;MigSupportSequence=True/False</td>
</tr>
<tr>
<td><strong>Migrate Oracle SQL using the latest (Beta version) tool</strong></td>
<td>• Migration Tool: application.properties&lt;br&gt;• Oracle Beta Configuration: oracle-migration-common.properties&lt;br&gt;oracle-migration-V1R8_330.properties</td>
<td>For Oracle (Beta) migration, the tool automatically matches the syntax and features supported by the GaussDB kernel based on the version parameters specified in user input commands. In this process, configuration files need no modification.</td>
</tr>
<tr>
<td><strong>Teradata Perl Migration</strong></td>
<td>• Migration Tool: application.properties&lt;br&gt;• Teradata: perl-migration.properties</td>
<td>add-timing-on=True/False&lt;br&gt;db-script-tagname=bteq/sql_lang/bteq,sql_lang&lt;br&gt;logging-level=error/warning/info&lt;br&gt;migrate-variables=True/False&lt;br&gt;remove-intermediate-files=True/False&lt;br&gt;target_files=overwrite/cancel&lt;br&gt;migrate-executequery=True/False</td>
</tr>
</tbody>
</table>

### 6.7.3 Preparations

Before the migration, create an input folder and an output folder, and copy all the SQL scripts to be migrated to the input folder. The following procedure describes how to prepare for the migration in Linux.

#### Step 1
Create an input folder and an output folder.

```bash
mkdir input
mkdir output
```

#### Step 2
Copy all Teradata/Oracle SQL scripts to be migrated to the input folder.

---End
If the encoding format of source files is not UTF-8, perform the following steps:

1. Open the `application.properties` file in the `config` folder.
2. Change the value of `encodingFormat` in the `application.properties` file to the required encoding format.

Migration Tool supports the UTF-8, ASCII, and GB2312 encoding formats. The values of `encodingFormat` are case-insensitive.

To obtain the encoding format of a source file in Linux, run the following command on the server where the source file is located:

```
file -bi <input file name>
```

### 6.7.4 Executing Migration Tool

**Precautions**

- Before starting Migration Tool programs, specify the path of the output folder. Separate the input folder path, output folder path, and log path with spaces. The path of the input folder cannot contain spaces. Spaces in a path will cause an error in Migration Tool execution. For details, see Troubleshooting.

- If the output folder contains subfolders or files, Migration Tool deletes the subfolders and files or overwrites them based on parameter settings in the `application.properties` configuration file in the `config` folder before the migration. Deleted or overwritten subfolders and files cannot be restored by Migration Tool.

- If migration is performed concurrently on the same server (executed by the same or different Migration Tools), different migration tasks must use different output folder paths and log paths.

- You can specify a log path by specifying optional parameters. If the path is not specified, Migration Tool automatically creates a log folder under `TOOL_HOME`. For details, see Log Reference.

**Migration Methods**

You can execute `migrate.sh` in Linux or `migrationtool.jar` in Windows to perform migration. For details, see Table 6-17 and Table 6-18.

<table>
<thead>
<tr>
<th>Table 6-17 Migration in Linux (migrate.sh)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scenario</strong></td>
</tr>
<tr>
<td>Teradata SQL Migration</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
### Table 6-18 Migration in Windows (migrationtool.jar)

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Command-Line Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Teradata SQL Migration</strong></td>
<td><code>&gt; java -jar migrationtool.jar</code></td>
</tr>
<tr>
<td></td>
<td><code>--source-db Teradata</code></td>
</tr>
<tr>
<td></td>
<td><code>--application-lang SQL</code></td>
</tr>
<tr>
<td></td>
<td><code>--input-folder &lt;input-script-path&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>--output-folder &lt;output-script-path&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>[-log-folder &lt;log-path&gt;]</code></td>
</tr>
<tr>
<td><strong>Oracle SQL Migration</strong></td>
<td><code>&gt; java -jar migrationtool.jar</code></td>
</tr>
<tr>
<td></td>
<td><code>--source-db Oracle</code></td>
</tr>
<tr>
<td></td>
<td><code>--application-lang SQL</code></td>
</tr>
<tr>
<td></td>
<td><code>--input-folder &lt;input-script-path&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>--output-folder &lt;output-script-path&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>[-log-folder &lt;log-path&gt;]</code></td>
</tr>
<tr>
<td><strong>Migrate Oracle SQL using the latest (Beta version) tool</strong></td>
<td><code>&gt; java -jar migrationtool.jar</code></td>
</tr>
<tr>
<td></td>
<td><code>--source-db OracleBeta</code></td>
</tr>
<tr>
<td></td>
<td><code>--application-lang SQL</code></td>
</tr>
<tr>
<td></td>
<td><code>--input-folder &lt;input-script-path&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>--output-folder &lt;output-script-path&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>[-log-folder &lt;log-path&gt;]</code></td>
</tr>
<tr>
<td><strong>Teradata Perl Migration</strong></td>
<td><code>&gt; java -jar migrationtool.jar</code></td>
</tr>
<tr>
<td></td>
<td><code>--source-db Teradata</code></td>
</tr>
<tr>
<td></td>
<td><code>--application-lang Perl</code></td>
</tr>
<tr>
<td></td>
<td><code>--input-folder &lt;input-script-path&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>--output-folder &lt;output-script-path&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>[-log-folder &lt;log-path&gt;]</code></td>
</tr>
</tbody>
</table>
### Command parameter description:

- **source-db**: specifies the source database. The value can be **Teradata**, **Oracle**, or **OracleBeta**, which is case-insensitive.

- **migration-type**: specifies the migration type. This parameter is optional. Migration Tool supports the following migration types:
  - **Bulk**: Migrate DML and DDL scripts.
  - **BLogic**: Migrate service logic, such as procedures and functions. **BLogic** is used only for Oracle PL/SQL.

### Command output description:

- **Migration process start time** indicates the migration start time and **Migration process end time** indicates the migration end time. **Total process time** indicates the total migration duration, in milliseconds. In addition, the total number of migrated files, total number of processors, number of used processors, log file path, and error log file path are also displayed on the console.

- For details, see [Migrate](migrationtool.jar).

### Task Example

- **Example for Linux**: Run the following command in Linux to migrate the SQL files of the Oracle database to the SQL scripts applicable to and DWS:

  ```sh
  sh migrate.sh Oracle --input-folder /home/testmigration/Documentation/input --output-folder /home/testmigration/Documentation/output --log-folder /home/testmigration/Documentation --migration-type bulk
  ```

  Migration details are displayed on the console (including the progress and completion status):

  ```
  Migration: MigrationTool version initiated by User1
  Migration process start time : Tue Aug 08 16:08:06 GMT+08:00 2017
  Number of available processors : 4 Configured simultaneous processes in the Tool : 3
  33% completed
  66% completed
  100% completed
  **************************************************************************
  Total number of SQL files in input folder : 3
  Number of queries failed : 0
  Number of files failed : 0
  Number of files migrated successfully : 3
  **************************************************************************
  Log file : migrationtool.log is placed in the path : /home/migrationtool/example/log
  Error Log file : migrationtoolError.log is placed in the path : /home/migrationtool/example/log
  Migration process end time : Tue Aug 08 16:08:08 GMT+08:00 2017
  Total process time : 2680 ms
  ```

- **Example for Windows**: Run the following command to migrate the SQL files of the Oracle database to the SQL scripts applicable to DWS:

  ```java
  java -jar migrationtool.jar --source-db Oracle --input-folder D:\test\migration\input --output-folder D:\test\migration\output --log-folder D:\test\migration\log --migration-type bulk
  ```

  Migration details are displayed on the console (including the progress and completion status):

  ```
  Migration: MigrationTool version initiated by User1
  Migration process start time : Tue Aug 08 16:08:06 GMT+08:00 2017
  Number of available processors : 4 Configured simultaneous processes in the Tool : 3
  33% completed
  66% completed
  100% completed
  **************************************************************************
  Total number of SQL files in input folder : 3
  ```
6.7.5 Viewing Output Files and Logs

Viewing and Verifying Output Files

After the migration is complete, you can use a comparison tool (for example, BeyondCompare®) to compare the output file with its input file. Input SQL files can also be formatted for easier comparison.

1. Run the following command in Linux and view output files in the output folder: Operations in Windows are not described here.
   ```
   cd OUTPUT
   ls
   ```

   Information similar to the following is displayed:

   ```
   formattedSource  output
   user1@node79:~/Documentation/MigrationTool/OUTPUT> cd output
   user1@node79:~/Documentation/MigrationTool/OUTPUT/output> ls
   in_index.sql    input.sql    Input_table.sql    in_view.sql    MetadataInput.sql
   ```

2. Use the comparison tool to compare the output file with its input file. Check whether the keywords in the migrated SQL file meet the requirements of the target database.

Viewing Log Files

Execution information and error messages are written into corresponding log files. For details, see Log Reference.

Check whether errors are logged. If they are, rectify the faults by following the instructions in Troubleshooting.

6.7.6 Troubleshooting

Migration related issues can be classified into:

- Tool execution issues: No output or incorrect output is displayed because Migration Tool partially or fully failed to execute. For details, see Troubleshooting and FAQs.
- Migration syntax issues: Migration Tool did not correctly recognize or migrate the migration syntax. For details, see Constraints and Limitations.

6.7.7 Teradata SQL Migration

Migration Tool supports the migration from Teradata to DWS, including the migration of schemas, DML, queries, system functions, and type casting.
Teradata SQL Migration in Linux

Run the following commands to set the source database, input and output folder paths, log paths, and application language:

```
sh migrate.sh
  --source-db Teradata
  --input-folder <input-script-path>
  --output-folder <output-script-path>
  --log-folder <log-path>
  --application-lang SQL
```

For example:

```
sh migrate.sh --source-db Teradata --input-folder /home/migrationtool/example/input --output-folder /home/migrationtool/example/output --log-folder /home/migrationtool/example/log --application-lang SQL
```

During the execution of Migration Tool, the migration summary, including the progress and completion status, is displayed on the console. Execution information and errors are written into 8.2 SQL Migration Logs.

```
Migration: MigrationTool version initiated by User1
Migration process start time : Tue Aug 08 16:08:06 GMT+08:00 2017
Number of available processors : 4
Configured simultaneous processes in the Tool : 3
11% completed
22% completed
33% completed
44% completed
55% completed
66% completed
77% completed
88% completed
100% completed
**************************************************************************
Started post processing
Completed post processing
**************************************************************************
Total number of SQL files in input folder : 9
Number of queries failed : 0
Number of files failed : 0
Number of files migrated successfully : 9
**************************************************************************
Log file: migrationtool.log is placed in the path : ./log
Error Log file: migrationtoolError.log is placed in the path : ./log
Migration process end time : Tue Aug 08 16:08:08 GMT+08:00 2017
Total process time : 2680 ms
```

Teradata SQL Migration in Windows

Run the following commands to set the source database, input and output folder paths, log paths, and application language:

```
java -jar migrationtool.jar
  --source-db Teradata
  --input-folder <input-script-path>
  --output-folder <output-script-path>
  --log-folder <log-path>
  --application-lang SQL
```

For example:

```
java -jar migrationtool.jar --source-db Teradata --input-folder D:\test\migration\input --output-folder D:\test\migration\output --log-folder D:\test\migration\log --application-lang SQL
```
During the execution of Migration Tool, the migration summary, including the progress and completion status, is displayed on the console. Execution information and errors are written into log files.

Migration: MigrationTool version initiated by User1
Migration process start time : Tue Aug 08 16:08:06 GMT+08:00 2017
Number of available processors : 4
Configured simultaneous processes in the Tool : 3
11% completed
22% completed
33% completed
44% completed
55% completed
66% completed
77% completed
88% completed
100% completed
**************************************************************************
Started post processing
Completed post processing
**************************************************************************
Total number of SQL files in input folder : 9
Number of queries failed : 0
Number of files failed : 0
Number of files migrated successfully : 9
**************************************************************************
Log file : migrationtool.log is placed in the path : ./log
Error Log file : migrationtoolError.log is placed in the path : ./log
Migration process end time : Tue Aug 08 16:08:08 GMT+08:00 2017
Total process time : 2680 ms

For details about how to migrate Teradata SQL using Migration Tool, see Executing Migration Tool.

Best Practices

To optimize the migration, you are advised to follow the standard practices for inputs files.

- Avoid extra parentheses: Using extra parentheses in a query may cause unexpected migration or partial migration of scripts.
  ```sql
  ..
  IN (((((SELECT NULL(BYTEINT) FROM categories1))))) ORDER BY category_name;
  ```

- Avoid extra comments: Using extra comments in the query may cause unexpected migration or partial migration of scripts.
  ```sql
  -- Delete the records
  DELETE
  FROM --Use one table
  tblName --Case sensitive names
  WHERE <condition>;
  -- Completed deletion
  ```

6.8 Oracle SQL Migration

Migration Tool supports the migration from Oracle to DWS, including the migration of schemas, DML, queries, system functions, and PL/SQL.

Oracle SQL Migration in Linux

Run the following commands to set the source database, input and output folder paths, log paths, application language, and migration type:
When migrating common DDL statements (tables, views, indexes, sequences, and so on) that do not contain PL/SQL statements, use the Bulk mode (that is, set **migration-type** to **Bulk**). When migrating objects such as functions, procedures, and packages that contain PL/SQL statements, use the BLogic mode (that is, set **migration-type** to **BLogic**).

In addition, common DDL scripts and PL/SQL scripts should be placed in different input folders for migration.

Example of the command with sample folder information and **migration-type** set as **Bulk**:
```
sh migrate.sh --source-db Oracle --input-folder /home/migrationtool/example/input --output-folder /home/migrationtool/example/output --log-folder /home/migrationtool/example/log --application-lang SQL --migration-type bulk
```

**Oracle SQL Migration in Windows**

Run the following commands to set the source database, input and output folder paths, log paths, application language, and migration type:
```
java -jar migrationtool.jar --source-db Oracle --input-folder <input-script-path> --output-folder <output-script-path> --log-folder <log-path> --application-lang Oracle --migration-type <migration-type>
```

- When migrating common DDL statements (tables, views, indexes, sequences, and so on) that do not contain PL/SQL statements, use the Bulk mode (that is, set **migration-type** to **Bulk**).

Example of the command with sample folder information and **migration-type** set as **Bulk**:
```
java -jar migrationtool.jar --source-db Oracle --input-folder D:\test\migration\input --output-folder D:\test\migration\output --log-folder D:\test\migration\log --application-lang SQL --migration-type bulk
```

During the execution of Migration Tool, the migration summary, including the progress and completion status, is displayed on the console. Execution information and errors are written into **log files**.

Migration: MigrationTool version initiated by User1
Migration process start time : Tue Aug 08 16:08:06 GMT+08:00 2017
Number of available processors : 4
Configured simultaneous processes in the Tool : 3
11% completed
22% completed
33% completed
44% completed
55% completed
66% completed
77% completed
88% completed
100% completed
**************************************************************************
Total number of SQL files in input folder : 9
Number of queries failed : 0
Number of files failed : 0
Number of files migrated successfully : 9

Log file : migrationtool.log is placed in the path : ./log
Error Log file : migrationtoolError.log is placed in the path : ./log
Migration process end time : Tue Aug 08 16:08:09 GMT+08:00 2017
Total process time : 680 ms

- When migrating objects such as functions, procedures, and packages that contain PL/SQL statements, use the BLogic mode (that is, set migration-type to BLogic).

Example of the command with sample folder information and migration-type set as BLogic:
```
java -jar migrationtool.jar --source-db Oracle --input-folder D:\test\migration\input --output-folder D:\test\migration\output --log-folder D:\test\migration\log --application-lang SQL --migration-type blogic
```

During the execution of Migration Tool, the migration summary, including the progress and completion status, is displayed on the console. Execution information and errors are written into log files.

Migration: MigrationTool version initiated by User1
Migration process start time : Tue Aug 08 16:08:08 GMT+08:00 2017
Number of available processors : 4
Configured simultaneous processes in the Tool : 3
11% completed
22% completed
33% completed
44% completed
55% completed
66% completed
77% completed
88% completed
100% completed

Note that common DDL scripts and PL/SQL scripts should be placed in different input folders for migration.

**Oracle PACKAGE Migration Precautions**

1. The package specifications (that is, the package header) and the package body should be placed in different files and in the same input path for migration.

2. You need to migrate common DDL statements (including all table structure information referenced in the PACKAGE script) in Bulk mode to form a dictionary in the config/create-types-UDT.properties file. Then, migrate the package specification (that is, the package header) and the package body in BLogic mode. The details are as follows:

   When some Oracle PACKAGE defines package specifications, the tbName.colName %TYPE syntax is used to declare custom record types based on other table object
Example

```
CREATE OR REPLACE PACKAGE p_emp
AS
  --Define the RECORD type
  TYPE re_emp IS RECORD(
    rno emp.empno%TYPE,
    rname emp.empname%TYPE
  );
END;
```

The GaussDB database of the current kernel version does not support the `tbName.colName%TYPE` syntax. Therefore, during the migration, the Migration Tool needs to build a database context environment containing the `emp` table information. In this case, you need to use Migration Tool to migrate all table creation scripts (that is, use the Bulk mode to migrate common DDL statements). Migration Tool automatically generates corresponding data dictionaries. After the context environment containing various table information is built, the Blogic mode can be used to migrate the Oracle PACKAGE. In this case, the `re_emp` record type is migrated according to the column type of the `emp` table.

Expected output

```
CREATE TYPE p_emp.re_emp AS (
  rno NUMBER(4),
  rname VARCHAR2(10)
);
```

For details about how to migrate Oracle SQL using Migration Tool (Beta), see Executing Migration Tool.

### 6.9 Oracle (Beta) SQL Migration

Oracle (Beta) is a beta version of the latest Oracle Migration Tool. It provides enhanced parsing and migration capabilities.

Migration Tool (Beta) supports the migration from Oracle to DWS, including the migration of schemas, DML, queries, system functions, and PL/SQL.

**Oracle (Beta) SQL Migration in Linux**

Run the following commands to set the source database, input and output folder paths, log paths:

```
sh migrate.sh
  --source-db Oraclebeta
  --input-folder <input-script-path>
  --output-folder <output-script-path>
  --log-folder <log-path>
  --version-number <Gauss Kernel Version>
  --application-lang SQL
```

**Oracle (Beta) SQL Migration in Windows**

Run the following commands to set the source database, input and output folder paths, log paths, and application language:

```
java -jar migrationtool.jar
  --source-db Oraclebeta
  --input-folder <input-script-path>
  --output-folder <output-script-path>
  --log-folder <log-path>
```

Issue 03 (2019-08-08) Copyright © Huawei Technologies Co., Ltd. 372
--version-number <Gauss Kernel Version>
--application-lang SQL

During the execution of Migration Tool, the migration summary, including the progress and completion status, is displayed on the console. Execution information and errors are written into log files.

Migration: MigrationTool version initiated by User1  
Migration process start time : Tue Aug 08 16:08:06 GMT+08:00 2017  
Number of available processors : 4  
Configured simultaneous processes in the Tool : 3  
33% completed  
66% completed  
100% completed  
**************************************************************************  
Total number of SQL files in input folder : 3  
Number of queries failed : 0  
Number of files failed : 0  
Number of files migrated successfully : 3  
**************************************************************************  
Log file : migrationtool.log is placed in the path : /home/migrationtool/example/log  
Error Log file : migrationtoolError.log is placed in the path : /home/testmigration/example/log  
Migration process end time : Tue Aug 08 16:08:08 GMT+08:00 2017  
Total process time : 2680 ms

For details about how to migrate Oracle SQL using Migration Tool (Beta), see Executing Migration Tool.

### 6.10 Teradata Perl Migration

#### Overview

This section describes details about the migration of Teradata Perl files.

Use `migrate.sh` or `migration.jar` with `--application-lang=perl` to migrate Teradata BTEQ or SQL_LANG scripts within Perl files to DWS compatible with Perl files. After migrating Perl files, you can verify the migration by comparing the output file with its input file using a comparison tool.

The process of migrating Perl files is as follows:

1. Perform the operations in Prerequisites.
2. Create an input folder and copy the Perl files to be migrated to the folder. For example, create a `/migrationfiles/perlfiles` input folder.
3. Execute the Perl Migration Tool for Perl and set `db-script-tag-name` to BTEQ or SQL_LANG.
   a. The Perl Migration Tool extracts the BTEQ or SQL_LANG scripts from the Perl files.
   b. The Perl Migration Tool invokes the Teradata SQL Migration Tool to migrate the extracted SQL scripts. For details about Teradata SQL migration, see Teradata Syntax Migration.
   c. Perl files embed back the migrated scripts.
4. Migration Tool creates the migrated files in the specified output folder. If no output folder is specified, Migration Tool creates an output folder named converted in the input folder, for example, `/migrationfiles/perlfiles/converted`.
Perl variables containing SQL statements can also be migrated to SQL by setting the `migrate-variables` parameter.

For perl v 5.10.0 and above are compatible.

Teradata Perl Migration in Linux

To migrate Perl files, execute Migration Tool (`migrate.sh`) with `--source-db Teradata` and `--application-lang Perl` parameter values. Migration Tool supports the migration of BTEQ and SQL_LANG scripts. You can specify the scripts to be migrated by setting `db-script-tag-name`.

Run the following commands to set the source database, input and output folder paths, log paths, and application language:

```
sh migrate.sh
--source-db|-S Teradata
--application-lang|-A Perl
--input-folder|-I <input-script-path>
--output-folder|-O <output-script-path>
--migration-type|-M <Bulk or BLogic>
[--log-folder|-L <log-path>]
```

Teradata Perl Migration in Windows

To migrate Perl files, execute Migration Tool (`migrationtool.jar`) with `--source-db Teradata` and `--application-lang Perl` parameter values. Migration Tool supports the migration of BTEQ and SQL_LANG scripts. You can specify the scripts to be migrated by setting `db-script-tag-name`.

Run the following commands to set the source database, input and output folder paths, log paths, and application language:

```
java -jar migrationtool.jar
--source-db|-S Teradata
--application-lang|-A Perl
--input-folder|-I <input-script-path>
--output-folder|-O <output-script-path>
--migration-type|-M <Bulk or BLogic>
[--log-folder|-L <log-path>]
```

For example:

```
java -jar migrationtool.jar -I ../executequery/ -O output/ -S teradata -M Bulk -A PERL
```

During the execution of Migration Tool, the migration summary, including the progress and completion status, is displayed on the console.

```
Stage 1/3: Extracting SQL contents from perl files.
Extraction completed
Stage 2/3: Migrating SQL contents
Migration completed
Stage 3/3: Merging SQL contents.
Merging completed
**************************************************************************
Number of Perl files in input directory : 11
Number of Queries failed : 0
Number of SQL files failed : 0
Number of Perl file failures while extracting sql contents : 0
Number of Perl file failures while merging sql contents : 0
Number of Perl directory failures while extracting sql contents : 0
Number of Perl directory failures while merging sql contents : 0
Number of Perl files migrated successfully : 11
```
The following is an example of the summary shown in the case of a failure:

Stage 1/3: Extracting SQL contents from perl files.
Extraction completed
Stage 2/3: Migrating SQL contents
Migration completed
Stage 3/3: Merging SQL contents.
Merging completed

Number of Perl files in input directory : 11
Number of Queries failed : 0
Number of SQL files failed : 0
Number of Perl file failures while extracting sql contents : 0
Number of Perl file failures while merging sql contents : 1
Number of Perl directory failures while extracting sql contents : 0
Number of Perl directory failures while merging sql contents : 0

For details about the parameters for Teradata Perl migration, see Teradata Perl Configuration.

For details about command line parameters, see Migrate (migrationtool.jar).
NOTE

- Migration Tool formats the input and output files and saves them in the output folder. You can compare the formatted input files with the output files.
- Ensure that there are no spaces in the input path. If there is a space, Migration Tool throws an error. For details, see Troubleshooting.
- For details about logs, see Log Reference.
- If the output folder contains subfolders or files, Migration Tool deletes the subfolders and files or overwrites them based on parameter settings in the application.properties configuration file in the config folder before the migration. Deleted or overwritten subfolders and files cannot be restored by Migration Tool.
- Process start time indicates the migration start time and Process end time indicates the migration end time. Process total time indicates the total migration duration, in milliseconds. In addition, the total number of migrated files, total number of processors, number of used processors, log file path, and error log file path are also displayed on the console.
- Set --add-timing-on to true in the perl-migration.properties file to add a custom script to calculate statement execution time.

Example:

**Input**
```
$V_SQL2 = "SELECT T1.userTypeInd FROM T07_EBM_CAMP T1 WHERE T1.Camp_List_Id = "$abc";"
$STH = $dbh->prepare($V_SQL2);
$sth->execute();
@rows = $sth->fetchrow();
```

**Output**
```
$V_SQL2 = "SELECT T1.userTypeInd FROM T07_EBM_CAMP T1 WHERE T1.Camp_List_Id = "$abc";"
$STH = $dbh->prepare($V_SQL2);
use Time::HiRes qw/gettimeofday/;
my $start = [Time::HiRes::gettimeofday()];
$sth->execute();
my $elapsed = Time::HiRes::tv_interval($start);
$elapsed = $elapsed * 1000;
printf("Time: %.3f ms\n", $elapsed);
@rows = $sth->fetchrow();
```

- GROUP and OTHERS must not have write permission for the files or folders specified by --input-folder. That is, the privilege for the folder specified by --input-folder must not be higher than 755. For security purposes, Migration Tool will not be executed if the input files or folders have the write permission.
- If migration tasks are executed concurrently, the input folder must be unique for each task.

Best Practices

To optimize the migration, you are advised to follow the standard practices:

- **BTEQ** scripts must be in the following format:
  
  ```
  print BTEQ <<ENDOFINPUT;
  TRUNCATE TABLE employee;
  ENDOFINPUT
  close(BTEQ);
  ```

- **SQL_LANG** scripts must be in the following format:
  
  ```
  my $SQL=<<SQL_LANG;
  TRUNCATE TABLE employee;
  SQL_LANG
  ```

- Comment must not contain the following information:
  - print BTEQ <<ENDOFINPUT
  - ENDOFINPUT
6.11 SQL Formatter

The SQL formatter is provided to improve the readability of the SQL. It formats the SQL statements by adding/removing lines and contextual level indentation. The SQL Formatter is applied to the migrated output and can be applied to the input files.

Use the `formattedSourceRequired` parameter to enable/disable the SQL formatter for the source SQL files. If set to true, a copy of the input files is formatted and saved to `{outputpath}/formattedSource` folder.

The SQL formatter is supported for Teradata SQL Migration, and the Oracle SQL Migration. SQL scripts in Teradata Perl file migration are also formatted. The Oracle (Beta) tool does not include the SQL Formatter.

Input: SQL FORMATTER

```sql
select p1.parti_encode ,p1.accting_type_cd ,p1.prod_code ,p1.cust_type_cd ,p1.accting_amt_type_cd ,p1.accting_num_1 ,p1.accting_num_2 ,p1.accting_num_3 ,p1.accting_num_4 ,p1.accting_num_5 ,p1.accting_num_6 ,p1.start_dt ,p1.pre_effect_debit_gl_num ,p1.pre_effect_crdt_gl_num ,p1.after_effect_debit_gl_num ,p1.after_effect_crdt_gl_num ,coalesce( p1.start_dt ,cast( '30001231' as date format 'yyyymmdd' ) ) ,p1.accting_term ,p1.etl_job 
from ( 
    select rank ( start_dt ) as start_dt_id ,parti_encode ,accting_type_cd ,prod_code ,accting_amt_type_cd ,accting_num_1 ,accting_num_2 ,accting_num_3 ,accting_num_4 ,accting_num_5 ,accting_num_6 ,start_dt ,pre_effect_debit_gl_num ,pre_effect_crdt_gl_num ,after_effect_debit_gl_num ,after_effect_crdt_gl_num ,accting_term ,etl_job 
    from ccting_subj_para_h_mf0_a_cur_i 
) p1 
left join ( 
    select rank ( start_dt ) - 1 as start_dt_id ,parti_encode ,accting_type_cd ,prod_code ,accting_amt_type_cd ,accting_num_1 ,accting_num_2 ,accting_num_3 ,accting_num_4 ,accting_num_5 ,accting_num_6 ,start_dt ,accting_term 
    from ccting_subj_para_h_mf0_a_cur_i 
) p2 
    on p1.start_dt_id = p2.start_dt_id 
    and p1.parti_encode = p2.parti_encode 
    and p1.accting_type_cd = p2.accting_type_cd 
    and p1.prod_code = p2.prod_code 
    and p1.cust_type_cd = p2.cust_type_cd 
    and p1.accting_amt_type_cd = p2.accting_amt_type_cd 
    and p1.accting_num_1 = p2.accting_num_1 
    and p1.accting_num_2 = p2.accting_num_2 
    and p1.accting_num_3 = p2.accting_num_3 
    and p1.accting_num_4 = p2.accting_num_4 
    and p1.accting_num_5 = p2.accting_num_5 
    and p1.accting_num_6 = p2.accting_num_6 
    and p1.accting_term = p2.accting_term ;
```

Output

```sql
SELECT p1.parti_encode ,p1.accting_type_cd ,p1.prod_code ,p1.cust_type_cd ,p1.accting_amt_type_cd ;
```
FROM (
  SELECT rank (
    ) over ( ORDER BY start_dt ASC ) AS start_dt_id
   ,parti_encode
   ,accting_type_cd
   ,prod_code
   ,cust_type_cd
   ,accting_amt_type_cd
   ,accting_num_1
   ,accting_num_2
   ,accting_num_3
   ,accting_num_4
   ,accting_num_5
   ,accting_num_6
   ,start_dt
   ,pre_effect_debit_gl_num
   ,pre_effect_crdt_gl_num
   ,after_effect_debit_gl_num
   ,after_effect_crdt_gl_num
   ,accting_term
   ,etl_job
FROM ccting_subj_para_h_mf0_a_cur_i
) p1 LEFT JOIN (  
  SELECT rank (  
    ) over ( ORDER BY start_dt ASC ) - 1 AS start_dt_id  
   ,parti_encode  
   ,accting_type_cd  
   ,prod_code  
   ,cust_type_cd  
   ,accting_amt_type_cd  
   ,accting_num_1  
   ,accting_num_2  
   ,accting_num_3  
   ,accting_num_4  
   ,accting_num_5  
   ,accting_num_6  
   ,start_dt  
   ,etl_job  
FROM ccting_subj_para_h_mf0_a_cur_i  
) p2  
ON p1.start_dt_id = p2.start_dt_id  
AND p1.parti_encode = p2.parti_encode  
AND p1.accting_type_cd = p2.accting_type_cd  
AND p1.prod_code = p2.prod_code  
AND p1.cust_type_cd = p2.cust_type_cd  
AND p1.accting_amt_type_cd = p2.accting_amt_type_cd  
AND p1.accting_num_1 = p2.accting_num_1  
AND p1.accting_num_2 = p2.accting_num_2  
AND p1.accting_num_3 = p2.accting_num_3  
AND p1.accting_num_4 = p2.accting_num_4  
AND p1.accting_num_5 = p2.accting_num_5  
AND p1.accting_num_6 = p2.accting_num_6  
AND p1.accting_term = p2.accting_term  
,
) rdt_gl_num
,

COALESCE( p2.start_dt ,CAST( '30001231' AS DATE ) )
6.12 Teradata Syntax Migration

6.12.1 Overview

The Teradata Migration Syntax lists the Teradata features supported by the Syntax Migration tool, and for each feature provides the Teradata syntax and the equivalent GaussDB 100 syntax. The syntax listed in this section illustrates the internal migration logic used to migrate the Teradata scripts.

It is also a reference for the database migration team and for customer site verification of Teradata scripts migration.

6.12.2 Schema Objects

6.12.2.1 Overview

This section contains the migration syntax for migrating Teradata schema objects. The migration syntax decides how the supported keywords/features are migrated.

Database Schema refers to the organization of data within a database. Migration Tool facilitates schema migration from Teradata to DWS.

6.12.2.2 Table Migration

Overview

The keyword MULTISET VOLATILE specific to table is provided in the input file, and the keyword is unsupported in DWS. Therefore, the tool replaces it with the LOCAL TEMPORARY/UNLOGGED keyword during the migration process. Use the session_mode configuration parameter to set the default table type (SET/ MULTISET) for CREATE TABLE.

CREATE TABLE

The Teradata CREATE TABLE (short key CT) statements are used to create new tables.

Example: Input: CREATE TABLE

```sql
CT tab1 ( id INT );
```

Output

```sql
CREATE TABLE tab1 ( id INTEGER
```
When using `CREATE tab2 AS tab1`, a new table `tab2` is created with the structure copied from `tab1`. If the `CREATE TABLE` statement includes `WITH DATA` operator, then the data from `tab1` is also copied into `tab2`. When using `CREATE AS`, the behavior of the `CONSTRAINT` from the source table is retained in the new target table.

- If `session_mode` = Teradata, then the duplicate records in the target table must be removed. This is done by adding the `MINUS` operator in the migrated scripts.
- If `session_mode` = ANSI, then the duplicate records in the target table are allowed.

If the source table has a PRIMARY KEY or a UNIQUE CONSTRAINT, then it will not contain any duplicate records. In this case the MINUS operator is not required or added to remove duplicate records.

**Example: Input: CREATE TABLE AS with DATA (session_mode=Teradata)**

```
CREATE TABLE tab2
  AS tab1 WITH DATA;
```

**Output**

```
BEGIN
  CREATE TABLE tab2 (LIKE tab1 INCLUDING ALL EXCLUDING PARTITION EXCLUDING RELOPTIONS);
  INSERT INTO tab2
  SELECT * FROM tab1
  MINUS SELECT * FROM tab2;
END;
```

**Example: Input: CREATE TABLE AS with DATA AND STATISTICS**

```
CREATE SET VOLATILE TABLE tab2025
AS ( SELECT * from tab2023 )
WITH DATA AND STATISTICS
PRIMARY INDEX (LOGTYPE, OPERSEQ);
```

**Output**

```
CREATE LOCAL TEMPORARY TABLE tab2025
  DISTRIBUTE BY HASH ( LOGTYPE, OPERSEQ )
AS ( SELECT * FROM tab2023 );
ANALYZE tab2025;
```

**CHARACTER SET and CASESPECIFIC**

CHARACTER SET is used to specify the server character set for a character column. CASESPECIFIC specifies the case for character data comparisons and collations.
Use the `tdMigrateCharsetCase` configuration parameter to configure migration of CHARACTER SET and CASESPECIFIC. If `tdMigrateCharsetCase` is set to false, the tool will skip migration of the query and will log a message.

**Input (tdMigrateCharsetCase=True)**

```sql
CREATE MULTISET VOLATILE TABLE TAB1
  (  
    col1 INTEGER NOT NULL  
    ,col2 INTEGER NOT NULL  
    ,col3 VARCHAR(100) NOT NULL CHARACTER SET UNICODE CASESPECIFIC  
  )  
PRIMARY INDEX (col1,col2)  
ON COMMIT PRESERVE ROWS  
;
```

**Output**

```sql
CREATE LOCAL TEMPORARY TABLE TMP_RATING_SYS_PARA
  (  
    col1 INTEGER NOT NULL  
    ,col2 INTEGER NOT NULL  
    ,col3 VARCHAR(100) NOT NULL /* CHARACTER SET UNICODE CASESPECIFIC */  
  )  
DISTRIBUTE BY HASH (col1,col2)  
ON COMMIT PRESERVE ROWS  
;
```

**Input-Migration support for Character-based data type**

In Teradata, the below character sets support character-based length for string data types

- LATIN
- UNICODE
- GRAPHIC

However, the KANJISJIS character set support byte-based length for string data types. For example, `COLUMN_NAME VARCHAR(100) CHARACTER SET UNICODE CASESPECIFIC COLUMN_NAME VARCHAR(100) CHARACTER SET LATIN CASESPECIFIC` This can store up to 100 characters (not bytes).

So, if TD's LATIN, UNICODE and GRAPHIC character sets, VARCHAR should be migrated to NVARCHAR.

**Input**

```sql
CREATE TABLE tab1
  (  
    col1 VARCHAR(10),  
    COL2 CHAR(1)  
  );
```

**Output**

```sql
a) when default_charset = UNICODE/GRAPHIC
CREATE TABLE tab1 (  
  col1 NVARCHAR2 (10)  
  ,COL2 NVARCHAR2 (1)  
) ;
b) when default_charset = LATIN
```
CREATE TABLE tab1 (  
  col1 VARCHAR2 (10)  
, CO2 VARCHAR2 (1)  
) ;

**Input**

CREATE TABLE tab1  
(  
  col1 VARCHAR(10) CHARACTER SET UNICODE,  
  CO2 CHAR(1)  
) ;

**Output**

a) when default_charset = UNICODE/GRAPHIC  
CREATE TABLE  
  tab1 (  
  col1 NVARCHAR2 (10) /* CHARACTER SET UNICODE*/  
, CO2 NVARCHAR2 (1)  
  ) ;

b) when default_charset = LATIN  
CREATE TABLE  
  tab1 (  
  col1 NVARCHAR2 (10) /* CHARACTER SET UNICODE*/  
, CO2 CHAR(1)  
  ) ;

**VOLATILE**

The keyword VOLATILE specific to table is provided in the input file, and the keyword is unsupported in DWS. The tool replaces it with the **LOCAL TEMPORARY** keyword during the migration process. Volatile tables are migrated as local temporary or un-logged based on the configuration input.

**Input: CREATE VOLATILE TABLE**

CREATE VOLATILE TABLE T1 (c1 int , c2 int);

**Output**

CREATE LOCAL TEMPORARY TABLE T1 (  
  c1 INTEGER  
, c2 INTEGER  
) ;

**Input: CREATE VOLATILE TABLE AS WITH DATA** (session_mode=Teradata)

If the source table has a PRIMARY KEY or a UNIQUE CONSTRAINT, then it will not contain any duplicate records. In this case the MINUS operator is not required or added to remove duplicate records.

CREATE VOLATILE TABLE tabV1 (  
  C1 INTEGER DEFAULT 99  
, C2 INTEGER  
, C3 INTEGER  
, C4 NUMERIC (20,0) DEFAULT NULL (BIGINT)  
) PRIMARY INDEX (C1, C3) ;
CREATE TABLE tabV2 AS tabV1 WITH DATA PRIMARY INDEX (C1)
   ON COMMIT PRESERVE ROWS;

Output

CREATE LOCAL TEMPORARY TABLE tabV1 (  
   C1 INTEGER DEFAULT 99  
   ,C2 INTEGER  
   ,C3 INTEGER  
   ,C4 NUMERIC (20,0) DEFAULT CAST( NULL AS BIGINT )  
   ,CONSTRAINT XX1 PRIMARY KEY ( C1, C2 )  
) DISTRIBUTE BY HASH (C1);
BEGIN  
   CREATE TABLE tabV2 (  
      LIKE tabV1 INCLUDING ALL EXCLUDING PARTITION EXCLUDING REL OPTIONS EXCLUDING DISTRIBUTION  
      ) DISTRIBUTE BY HASH (C1);  
   INSERT INTO tabV2 SELECT * FROM tabV1;  
END  
/

SET

SET is a unique feature in Teradata. It does not allow duplicate records. It is addressed using the MINUS set operator. Migration Tool supports MULTISET and SET tables. SET table can be used with VOLATILE.

Input: SET TABLE

CREATE SET VOLATILE TABLE tab1 ... ;  
INSERT INTO tab1  
SELECT expr1, expr2, ...  
FROM tab1, ...  
WHERE ...,;

Output

CREATE LOCAL TEMPORARY TABLE tab1  
... ; INSERT INTO tab1  
SELECT expr1, expr2, ...  
FROM tab1, ...  
WHERE ....  
MINUS  
SELECT * FROM tab1 ;

MULTISET

MULTISET is a normal table, which is supported by all the DBs. Migration tool supports MULTISET and SET tables.

MULTISET table can be used with VOLATILE.

Input: CREATE MULTISET TABLE

CREATE VOLATILE MULTISET TABLE T1 (c1 int ,c2 int);

Output

CREATE  
LOCAL TEMPORARY TABLE  
T1 (  
   c1 INTEGER  
   ,c2 INTEGER  
   ... ; INSERT INTO tab1  
SELECT expr1, expr2, ...  
FROM tab1, ...  
WHERE ....  
MINUS  
SELECT * FROM tab1 ;
**TITLE**

The keyword **TITLE** is supported for Teradata Permanent, Global Temporary and Volatile tables. In the migration process, the TITLE text is migrated as a comment.

**NOTE**

If the TITLE text is split across multiple lines, then in the migrated query, the line breaks (ENTER) are replaced with a space.

Migration tool supports the TITLE tag in CREATE and SELECT statements.

**Input: CREATE TABLE with TITLE**

```
CREATE TABLE tab1 (
  c1  NUMBER(2) TITLE 'column_a'
);
```

**Output**

```
CREATE TABLE tab1 (
  c1  NUMBER(2) /* TITLE 'column_a' */
);
```

**Input: TABLE with multiline TITLE**

```
CREATE TABLE tab1 (
  c1  NUMBER(2) TITLE 'This is a very long title'
);
```

**Output**

```
CREATE TABLE tab1 (
  c1  NUMBER(2) /* TITLE 'This is a very long title' */
);
```

**Input: TABLE with COLUMN TITLE**

Migration Tool migrates COLUMN TITLE as a new outer query.

```
SELECT customer_id (TITLE 'cust_id')
FROM Customer_T
WHERE cust_id > 10;
```

**Output**

```
SELECT customer_id AS "cust_id"
FROM
  (SELECT
       customer_id
       FROM Customer_T
       WHERE cust_id > 10
  )
```

**Input: TABLE with COLUMN TITLE and QUALIFY**

Since qualify is migrated with an outer query, the same outer query is used for supporting the TITLE.
SELECT ord_id
    (TITLE 'Order_Id'), order_date, customer_id
FROM order_t
WHERE Order_Id > 100
QUALIFY ROW_NUMBER() OVER(PARTITION BY customer_id ORDER BY order_date DESC) <= 5;

Output

creation syntax
    "mig_tmp_alias1" AS "Order_Id"
FROM
    (SELECT
        ord_id AS "mig_tmp_alias1"
        ,ROW_NUMBER() OVER( PARTITION BY customer_id ORDER BY order_date DESC ) AS ROW_NUM1
    FROM
        order_t
    WHERE
        Order_Id > 100 ) Q1
WHERE
    Q1.ROW_NUM1 <= 5;

TITLE with ALIAS

If the TITLE is accompanied with an ALIAS, the tool will migrate it as follows:

- **TITLE with AS**: Tool will migrate it with the AS alias.
- **TITLE with NAMED**: Tool will migrate it with NAMED alias.
- **TITLE with NAMED and AS**: Tool will migrate it with AS alias.

Input: TABLE TITLE with NAMED and AS

SELECT  Acct_ID (TITLE 'Acc Code') (NAMED XYZ)  AS "Account Code"
    ,Acct_Name (TITLE 'Acc Name')
FROM    GT_JCB_01030_Acct_PBU
where "Account Code" > 500  group by "Account Code" ,Acct_Name ;

Output

SELECT  Acct_ID AS "Account Code"
    ,Acct_Name AS "Acc Name"
FROM    GT_JCB_01030_Acct_PBU
WHERE
    Acct_ID > 500
GROUP BY
    Acct_ID ,Acct_Name ;
**NOTE**

Currently the Migration tool supports the migration of the TITLE command included in the initial CREATE/ALTER statement. The subsequent references of the TITLE specified column are not supported. For example, in the CREATE TABLE statement below, the column `eid` with the TITLE Employee ID will be migrated to a comment but the reference of `eid` in the SELECT statement will be retained as it is.

Input

```sql
CREATE TABLE tab1 ( eid INT TITLE 'Employee ID');
SELECT eid FROM tab1;
```

Output

```sql
CREATE TABLE tab1 (eid INT /*TITLE 'Employee ID*/);
SELECT eid from tab1;
```

**INDEX**

The create table statement supports creation of an index. Migration tool supports the TABLE statement with PRIMARY INDEX and UNIQUE INDEX.

The tool will not add DISTRIBUT BY HASH creating a table with PRIMARY KEY and Non-Unique PRIMARY INDEX.

**Input: CREATE TABLE with INDEX**

```sql
CREATE SET TABLE DP_TEDW.B0381_ACCOUNT_OBTAINED_MAP,
    NO FALBACK, NO BEFORE JOURNAL,
    NO AFTER JOURNAL, CHECKSUM = DEFAULT
( Ranked_Id INTEGER NOT NULL
  , Source_System_Code SMALLINT NOT NULL
  , Operational_Acc Obtained_Id VARCHAR(100)
  , CHARACTER SET LATIN NOT CASESPECIFIC FORMAT 'X(50)'
  , Mapped_Id INTEGER NOT NULL
)  PRIMARY INDEX B0381_ACCOUNT_OBTAINED_idx_PR ( Ranked_Id )
UNIQUE INDEX B0381_ACCT_OBT_MAP__idx_SCD ( Source_System_Code )
INDEX B0381_ACCT_OBT_MAP__idx_OPID ( Operational_Acc Obtained_Id );
```

**Output**

```sql
CREATE TABLE DP_TEDW.B0381_ACCOUNT_OBTAINED_MAP
( Ranked_Id INTEGER NOT NULL
  , Source_System_Code SMALLINT NOT NULL
  , Operational_Acc Obtained_Id VARCHAR( 100 )
  , Mapped_Id INTEGER NOT NULL
)  DISTRIBUTE BY HASH ( Ranked_Id );
CREATE INDEX B0381_ACCT_OBT_MAP__idx_SCD ON DP_TEDW.B0381_ACCOUNT_OBTAINED_MAP
( Source_System_Code );
CREATE INDEX B0381_ACCT_OBT_MAP__idx_OPID ON DP_TEDW.B0381_ACCOUNT_OBTAINED_MAP
( Operational_Acc Obtained_Id );
```

**NOTE**

UNIQUE is removed in the index since index column list (organic_name) is not a super set of DISTRIBUT BY column list (serial_no, organic_name).

**Input - CREATE TABLE with Primary Key and Non-Unique Primary Index (DISTRIBUT BY HASH is not added)**

```sql
CREATE TABLE employee
    ( EMP_NO INTEGER
    , DEPT_NO INTEGER
```
### CONSTRAINT

A table CONSTRAINT is applied to multiple columns. Migration tool supports the following constraints:

- **CHECK constraint:** supported by DWS.
- **REFERENCES constraint / FOREIGN KEY:** migration currently NOT supported by tool.
- **PRIMARY KEY constraint:** migration supported by tool.
- **UNIQUE constraint:** migration supported by tool.

**Input:** CREATE TABLE with CONSTRAINT

```sql
CREATE TABLE DP_SEDW.T_170UT_HOLDER_ACCT,
NO FALLBACK,
NO BEFORE JOURNAL, NO AFTER JOURNAL
( BUSINESSDATE VARCHAR(10)
, SOURCESYSTEM VARCHAR(5)
, UPLOADCODE VARCHAR(1)
, HOLDER_NO VARCHAR(7) NOT NULL
, POSTAL_ADD_4 VARCHAR(40)
, EPF_IND CHAR(1)
, CONSTRAINT uq_t_170ut_hldr UNIQUE ( SOURCESYSTEM, UPLOADCODE, HOLDER_NO )
) PRIMARY INDEX ( HOLDER_NO, SOURCESYSTEM ) ;
```

**Output**

```sql
CREATE TABLE DP_SEDW.T_170UT_HOLDER_ACCT
( BUSINESSDATE VARCHAR(10)
, SOURCESYSTEM VARCHAR(5)
, UPLOADCODE VARCHAR(1)
, HOLDER_NO VARCHAR(7) NOT NULL
, POSTAL_ADD_4 VARCHAR(40)
, EPF_IND CHAR(1)
, CONSTRAINT uq_t_170ut_hldr UNIQUE ( SOURCESYSTEM, UPLOADCODE, HOLDER_NO )
) PRIMARY INDEX ( HOLDER_NO, SOURCESYSTEM ) ;
```

**Input**

After table creation, CONSTRAINT can be added to a table column to put some restriction at column level by using ALTER statement.
CREATE TABLE GCC_PLAN.T1033 ( ROLLOUT_PLAN_LINE_ID NUMBER NOT NULL,
    UDF_FIELD_VALUE_ID NUMBER NOT NULL ) ;

ALTER TABLE GCC_PLAN.T1033
ADD CONSTRAINT UDF_FIELD_VALUE_ID_PK UNIQUE (UDF_FIELD_VALUE_ID) ;

**Output**

CREATE TABLE GCC_PLAN.T1033 ( ROLLOUT_PLAN_LINE_ID NUMBER NOT NULL,
    UDF_FIELD_VALUE_ID NUMBER NOT NULL,
    CONSTRAINT UDF_FIELD_VALUE_ID_PK
    UNIQUE (UDF_FIELD_VALUE_ID) ) ;

**NOTE**

Need to put CONSTRAINT creation syntax inside table creation script after all column declaration.

**COLUMN STORE**

The table orientation can be converted from ROW-STORE to COLUMN store using the WITH (ORIENTATION=COLUMN) in the CREATE TABLE statement. This feature can be enabled/disabled using the rowstoreToColumnstore configuration parameter.

Migration tool supports the TITLE tag in CREATE and SELECT statements.

**Input: CREATE TABLE with change orientation to COLUMN STORE**

```sql
CREATE MULTISET VOLATILE TABLE tab1
    ( c1 VARCHAR(30) CHARACTER SET UNICODE,
      c2 DATE
    )
PRIMARY INDEX (c1, c2)
ON COMMIT PRESERVE ROWS;
```

**Output**

```sql
CREATE LOCAL TEMPORARY TABLE tab1
    ( c1 VARCHAR(30),
      c2 DATE
    ) WITH (ORIENTATION = COLUMN)
ON COMMIT PRESERVE ROWS
DISTRIBUTE BY HASH (c1, c2);
```

**PARTITION**

The tool does not support migration of partitions/subpartitions and the partition/subpartition keywords are commented in the migrated scripts:

- Range partition/subpartition
- List partition/subpartition
- Hash partition/subpartition

**Input - PARTITION BY RANGE_N**

```sql
CREATE
MULTISET VOLATILE TABLE
    tab1
    ,NO LOG ( Acct_Num VARCHAR( 60 )
    ,Tx_Dt DATE
    ,Tx_Tm CHAR( 10 )
```
```
CREATE LOCAL TEMPORARY TABLE tab1 (
  Acct_Num VARCHAR(60),
  Tx Dt DATE,
  Tx Tm CHAR(10),
  Tx Cd VARCHAR(30),
  Tx Amt DECIMAL(18,2)
) ON COMMIT PRESERVE ROWS DISTRIBUTE BY HASH (Acct_Num) /* PARTITION BY (RANGE_N(Tx Dt BETWEEN CAST('${TX_MONTH_START}' AS DATE FORMAT 'YYYYMMDD') AND CAST('${TX_DATE}', AS DATE FORMAT 'YYYYMMDD') EACH INTERVAL '1' DAY, NO RANGE OR UNKNOWN) */
```

**Output**

```
CREATE LOCAL TEMPORARY TABLE tab1
  (Acct_Num VARCHAR(60),
   Tx Dt DATE,
   Tx Tm CHAR(10),
   Tx Cd VARCHAR(30),
   Tx Amt DECIMAL(18,2)
) ON COMMIT PRESERVE ROWS DISTRIBUTE BY HASH (Acct_Num) /* PARTITION BY (RANGE_N(Tx Dt BETWEEN CAST('${TX_MONTH_START}' AS DATE FORMAT 'YYYYMMDD') AND CAST('${TX_DATE}', AS DATE FORMAT 'YYYYMMDD') EACH INTERVAL '1' DAY, NO RANGE OR UNKNOWN) */
```

**ANALYZE**

**Input - CREATE TABLE with INDEX**

CREATE TABLE EMP27 AS emp21 WITH DATA
  PRIMARY INDEX (EMPNO) ON COMMIT PRESERVE ROWS;

**Output**

```
BEGIN
CREATE TABLE EMP27 ( LIKE emp21 INCLUDING ALL EXCLUDING PARTITION EXCLUDING RELOPTIONS EXCLUDING DISTRIBUTION )
DISTRIBUTE BY HASH (EMPNO);
INSERT INTO EMP27
  select * from emp21;
END;
/
```

**Data Types**

The following data type mappings are supported by the migration tool:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>Numeric</td>
</tr>
<tr>
<td>BIGINT</td>
<td>BIGINT</td>
</tr>
<tr>
<td>BYTEINT</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>DECIMAL [(n,[m])]</td>
<td>DECIMAL [(n,[m])]</td>
</tr>
<tr>
<td>DOUBLE PRECISION</td>
<td>DOUBLE PRECISION</td>
</tr>
<tr>
<td>Input</td>
<td>Output</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>FLOAT</td>
<td>DOUBLE PRECISION</td>
</tr>
<tr>
<td>INT / INTEGER</td>
<td>INTEGER</td>
</tr>
<tr>
<td>NUMBER / NUMERIC</td>
<td>NUMERIC</td>
</tr>
<tr>
<td>NUMBER(n[,m])</td>
<td>NUMERIC (n[,m])</td>
</tr>
<tr>
<td>REAL</td>
<td>REAL</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>Character</td>
<td>Character</td>
</tr>
<tr>
<td>CHAR[(n)] / CHARACTER [(n)]</td>
<td>CHAR(n)</td>
</tr>
<tr>
<td>CLOB</td>
<td>CLOB</td>
</tr>
<tr>
<td>LONG VARCHAR</td>
<td>TEXT</td>
</tr>
<tr>
<td>VARCHAR(n) / CHAR VARYING(n) / CHARACTER VARYING(n)</td>
<td>VARCHAR(n)</td>
</tr>
<tr>
<td>DateTime</td>
<td>DateTime</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
</tr>
<tr>
<td>TIME [(n)]</td>
<td>TIME [(n)]</td>
</tr>
<tr>
<td>TIME [(n)] WITH TIME ZONE</td>
<td>TIME [(n)] WITH TIME ZONE</td>
</tr>
<tr>
<td>TIMESTAMP [(n)]</td>
<td>TIMESTAMP [(n)]</td>
</tr>
<tr>
<td>TIMESTAMP [(n)] WITH TIME ZONE</td>
<td>TIMESTAMP [(n)] WITH TIME ZONE</td>
</tr>
<tr>
<td>Period</td>
<td>Period</td>
</tr>
<tr>
<td>PERIOD(DATE)</td>
<td>daterange</td>
</tr>
<tr>
<td>PERIOD(TIME [(n)])</td>
<td>tsrange [(n)]</td>
</tr>
<tr>
<td>PERIOD(TIME WITH TIME ZONE)</td>
<td>tstzrange</td>
</tr>
<tr>
<td>PERIOD(TIMESTAMP [(n)])</td>
<td>tsrange [(n)]</td>
</tr>
<tr>
<td>PERIOD(TIMESTAMP WITH TIME ZONE)</td>
<td>tstzrange</td>
</tr>
<tr>
<td>Binary</td>
<td>Binary</td>
</tr>
<tr>
<td>BLOB[(n)]</td>
<td>blob</td>
</tr>
<tr>
<td>BYTE[(n)]</td>
<td>bytea</td>
</tr>
<tr>
<td>VARBYTE[(n)]</td>
<td>bytea</td>
</tr>
</tbody>
</table>
Support for Less Columns Specified

Migration tool supports queries that specify less number of columns (not all columns specified) during INSERT. This can happen when the input INSERT statement does not contain all the columns mentioned in the input CREATE statement. On migration, the columns are added with any default values specified.

⚠️ NOTE

This feature is supported if session_mode is Teradata.

- The SELECT statement for the INSERT-INTO-SELECT must not include the following:
  - Set operators
  - MERGE, TOP with PERCENT, TOP PERCENT with TIES

Input - TABLE with all columns of CREATE are not specified in the INSERT statement

CREATE VOLATILE TABLE
  Convert_Data3,
  NO LOG (zoneno CHAR(6), brno CHAR(6), currtype CHAR(4), Commuteno CHAR(4), Subcode CHAR(12), Accdate DATE format 'YYYY-MM-DD' NOT NULL, Acctime INTEGER, Quoteno CHAR(1), Quotedate DATE FORMAT 'YYYY-MM-DD', Tddrbal DECIMAL(18,0) DEFAULT 0, Tdcrbal DECIMAL(18,0), Tddramt DECIMAL(18,0) DEFAULT 25, Tdcramt DECIMAL(18,0), Tddrbal DECIMAL(18,2), Tdcrbal DECIMAL(18,2)) PRIMARY INDEX (BRNO, CURRTYPE, SUBCODE)
  ) ON COMMIT PRESERVE ROWS;

INSERT INTO Convert_Data3 (zoneno, brno, currtype, commuteno, subcode, accdate, acctime, quoteno, quotedate, tddrbal, tdcrbal)
  ) SELECT A.zoneno, A.brno, '014' currtype, '2' commuteno, A.subcode, A.Accdate, A.Acctime

CREATE LOCAL TEMPORARY TABLE Convert_Data3 ( 
  zoneno CHAR(6) ,
  brno CHAR(6) ,
  currtype CHAR(4) ,
  commuteno CHAR(4) ,
  subcode CHAR(12) ,
  accdate DATE NOT NULL ,
  acctime INTEGER ,
  quoteno CHAR(1) ,
  quotedate DATE ,
  lddral DECIMAL(18,0) DEFAULT 0 ,
  ldcrbal DECIMAL(18,0) ,
  tddramt DECIMAL(18,0) DEFAULT 25 ,
  tdcramt DECIMAL(18,0) ,
  tddrbal DECIMAL(18,2) ,
  tdcrbal DECIMAL(18,2) ) 
ON COMMIT PRESERVE ROWS DISTRIBUTE BY HASH ( 
  BRNO ,
  CURRTYPE ,
  SUBCODE 
) ;

INSERT INTO Convert_Data3 ( 
  lddral ,
  ldcrbal ,
  tddramt ,
  tdcramt ,
  zoneno ,
  brno ,
  currtype ,
  commuteno ,
  subcode ,
  accdate ,
  acctime ,
  quoteno ,
  quotedate ,
  tddrbal ,
  tdcrbal ) SELECT 
  0 ,
  NULL ,
  25 ,
  NULL ,
  A.zoneno ,
  A.brno ,
  '014' currtype ,
  '2' commuteno ,
  A.subcode ,
  A.Accdate ,
  A.Acctime ,
  '2' quoteno ,
  B.workdate quotedate
CAST( ( CAST( SUM ( CAST( A.tddrbal AS FLOAT ) * CAST( B.USCVRATE AS FLOAT ) ) AS FLOAT ) ) AS DECIMAL( 18,2 ) ) AS tddrbal,
CAST( ( CAST( SUM ( CAST( A.tdcrbal AS FLOAT ) * CAST( B.USCVRATE AS FLOAT ) ) AS FLOAT ) ) AS DECIMAL( 18,2 ) ) AS tdcrbal
FROM
table2 A MINUS SELECT
  lddrbal,
  ldcrbal,
  tddramt,
  tdcramt,
  zoneno,
  brno,
  currtype,
  commuteno,
  subcode,
  accdate,
  acctime,
  quoteno,
  quotedate,
  tddrbal,
  tdcrbal
FROM
  CONVERT_DATA3;

6.12.2.3 CHARACTER SET and CASESPECIFIC

CHARACTER SET is used to specify the server character set for a character column. CASESPECIFIC specifies the case for character data comparisons and collations.

Use the tdMigrateCharsetCase configuration parameter to configure migration of CHARACTER SET and CASESPECIFIC. If tdMigrateCharsetCase is set to false, the tool will skip migration of the query and will log a message.

Input (tdMigrateCharsetCase=True)

```
CREATE MULTISET VOLATILE TABLE TAB1
  (col1 INTEGER NOT NULL,
   col2 INTEGER NOT NULL,
   col3 VARCHAR(100) NOT NULL CHARACTER SET UNICODE CASESPECIFIC)
PRIMARY INDEX (col1,col2)
ON COMMIT PRESERVE ROWS;
```

Output

```
CREATE LOCAL TEMPORARY TABLE TMP_RATING_SYS_PARA
  (col1 INTEGER NOT NULL,
   col2 INTEGER NOT NULL,
   col3 VARCHAR(100) NOT NULL /* CHARACTER SET UNICODE CASESPECIFIC */) ON COMMIT PRESERVE ROWS
DISTRIBUTE BY HASH (col1,col2);
```

6.12.2.4 Index Migration

The order of column and table name for CREATE INDEX in Teradata is different from DWS. Use the configuration parameter `distributeByHash` to configure how the data is distributed across the cluster nodes. The tool will not add DISTRIBUTE BY HASH creating a table with Primary Key and Non-Unique Primary Index.
Input - Primary key is not superset of primary index and only one column is matched

```
CREATE TABLE good_5 (
    column_1 INTEGER NOT NULL PRIMARY KEY,
    column_2 INTEGER,
    column_3 INTEGER NOT NULL,
    column_4 INTEGER
) PRIMARY INDEX (column_1, column_2);
```

Output

```
CREATE TABLE good_5 (  
    column_1 INTEGER NOT NULL PRIMARY KEY,
    column_2 INTEGER,
    column_3 INTEGER NOT NULL,
    column_4 INTEGER
);  
```

Input - Primary key is not superset of primary index and no column is matched

```
CREATE SET TABLE DP_SEDWT.T_170UT_HOLDER_ACCT 
  NO FALLBACK
  NO BEFORE JOURNAL
  NO AFTER JOURNAL (  
    BUSINESSDATE VARCHAR(10)
    , SOURCESYSTEM VARCHAR(5)
    , UPLOADCODE VARCHAR(1)
    , HOLDER_NO VARCHAR(7) NOT NULL
    , POSTAL_ADD_4 VARCHAR(40)
    , EPF_IND CHAR(1)
    , PRIMARY KEY (UPLOADCODE, HOLDER_NO)
) PRIMARY INDEX (SOURCESYSTEM, EPF_IND);
```

Output

```
CREATE TABLE DP_SEDWT.T_170UT_HOLDER_ACCT (  
    BUSINESSDATE VARCHAR(10)
    , SOURCESYSTEM VARCHAR(5)
    , UPLOADCODE VARCHAR(1)
    , HOLDER_NO VARCHAR(7) NOT NULL
    , POSTAL_ADD_4 VARCHAR(40)
    , EPF_IND CHAR(1)
    , PRIMARY KEY (UPLOADCODE, HOLDER_NO)
);
```

Input - No primary key and unique index has index name

```
CREATE SET TABLE "DP_TEDW"."T0409_INTERNAL_ORG_GRP_FUNCT", 
  NO FALLBACK, NO BEFORE JOURNAL,
  NO AFTER JOURNAL (  
    Organization_Party_Id INTEGER NOT NULL
    , Function_Code SMALLINT NOT NULL
    , Intern_Funct_Strt_Date DATE FORMAT 'YYYY-MM-DD' NOT NULL
    , Intern_Funct_End_Date DATE FORMAT 'YYYY-MM-DD'
  )
  PRIMARY INDEX (Organization_Party_Id)
  UNIQUE INDEX ux_t0409_intr_fn_1 (Function_Code, Intern_Funct_Strt_Date)
  UNIQUE INDEX (Organization_Party_Id, Intern_Funct_Strt_Date);
```

Output

```
CREATE TABLE "DP_TEDW"."T0409_INTERNAL_ORG_GRP_FUNCT" (  
    Organization_Party_Id INTEGER NOT NULL
    , Function_Code SMALLINT NOT NULL
    , Intern_Funct_Strt_Date DATE NOT NULL
    , Intern_Funct_End_Date DATE
  )
```
CREATE INDEX ux_t0409_intr_fn_1 ON "DP_TEDW"."T0409_INTERNAL_ORG_GRP_FUNCT" ( Function_Code, Intern_Funct_Strt_Date );
CREATE UNIQUE INDEX ON "DP_TEDW"."T0409_INTERNAL_ORG_GRP_FUNCT" ( Organization_Party_Id, Intern_Funct_Strt_Date );

**Input - CREATE TABLE with Primary Key and Non-Unique Primary Index**
(DISTRIBUTE BY HASH is not added)

CREATE TABLE employee
(
    EMP_NO INTEGER,
    DEPT_NO INTEGER,
    FIRST_NAME VARCHAR(20),
    LAST_NAME CHAR(20),
    SALARY DECIMAL(10,2),
    ADDRESS VARCHAR(100),
    CONSTRAINT pk_emp PRIMARY KEY ( EMP_NO )
) PRIMARY INDEX ( DEPT_NO ) ;

**Output**

CREATE TABLE employee
(
    EMP_NO INTEGER,
    DEPT_NO INTEGER,
    FIRST_NAME VARCHAR(20),
    LAST_NAME CHAR(20),
    SALARY DECIMAL(10,2),
    ADDRESS VARCHAR(100),
    CONSTRAINT pk_emp PRIMARY KEY ( EMP_NO )
) ;

**6.12.2.5 View Migration**

CREATE VIEW (short key CV) is used with SELECT to create a new view.

**VIEW** keyword is supported in both Teradata and DWS, but the **SELECT** statements are enclosed in braces during the migration process. For more information, refer the following figures.

Use the **tdMigrateVIEWCHECKOPTION** configuration parameter to configure migration of views containing the **WITH CHECK OPTION** keyword. If tdMigrateVIEWCHECKOPTION is set to false, the tool will skip migration of the query and will log a message.

If the CREATE VIEW includes the **LOCK** keyword, then the migration of the VIEW query will be based on the value of **tdMigrateLOCKoption**.

**Input - CREATE VIEW**

CREATE VIEW DP_STEDW.MY_PARAM
AS
SELECT RUNDATE FROM DP_STEDW.DATE_TBL WHERE dummy = 1;

**Output**

CREATE OR REPLACE VIEW DP_STEDW.MY_PARAM
AS
SELECT RUNDATE FROM DP_STEDW.DATE_TBL WHERE dummy = 1;

**Input :CREATE VIEW WITH FORCE KEYWORD**
CREATE OR REPLACE FORCE VIEW IS2010_APP_INFO (APP_ID, APP_SHORTNAME, APP_CHNAME, APP_ENNAME) AS
  select t.app_id, t.app_shortname, t.app_chname, t.app_enname
from newdrms.seas_app_info t
WHERE t.app_status <> '2';

Output

CREATE OR REPLACE /*FORCE*/ VIEW IS2010_APP_INFO (APP_ID, APP_SHORTNAME, APP_CHNAME, APP_ENNAME) AS
  SELECT t.app_id, t.app_shortname, t.app_chname, t.app_enname
FROM newdrms.seas_app_info t
WHERE t.app_status <> '2';

REPLACE VIEW

In Teradata, REPLACE VIEW statement creates a new view, or re-creates the existing view. Migration Tool migrates this to the compatible CREATE OR REPLACE VIEW in DWS.

Input - REPLACE VIEW

REPLACE VIEW DP_STEDW.MY_PARAM AS SELECT RUNDATE
  FROM DP_STEDW.DATE_TBL
  WHERE dummy = 1;

Output

CREATE OR REPLACE VIEW DP_STEDW.MY_PARAM AS (SELECT RUNDATE
  FROM DP_STEDW.DATE_TBL
  WHERE dummy = 1);

Input - REPLACE RECURSIVE VIEW

Replace RECURSIVE VIEW reachable_from (emp_id,emp_name,DEPTH)
AS ( 
SELECT root.emp_id,root.emp_name,0 AS DEPTH 
FROM emp AS root 
WHERE root.mgr_id IS NULL);

**Output**

CREATE OR REPLACE VIEW reachable_from AS ( 
WITH RECURSIVE reachable_from ( 
emp_id,emp_name,DEPTH) 
AS ( 
SELECT root.emp_id,root.emp_name,0 AS DEPTH 
FROM emp AS root 
WHERE root.mgr_id IS NULL 
) SELECT * FROM reachable_from);

## CHECK OPTION

Use the `tdMigrateVIEWCHECKOPTION` configuration parameter to configure migration of views containing the **CHECK OPTION** keyword

If a view with **CHECK OPTION** is present in the source, then the **CHECK OPTION** is commented from the target database.

**Input - VIEW with CHECK OPTION**

```
CV  mgr15 AS SEL * 
FROM employee 
WHERE 
  manager_id = 15 WITH CHECK OPTION 
;
```

**Output (tdMigrateVIEWCHECKOPTION=True)**

```
CREATE OR REPLACE VIEW mgr15 AS ( 
SELECT * 
FROM employee 
WHERE 
  manager_id = 15 /*WITH CHECK OPTION */ 
) 
;
```

**Output (tdMigrateVIEWCHECKOPTION=False)**

```
CV  mgr15 AS SEL * 
FROM employee 
WHERE 
  manager_id = 15 WITH CHECK OPTION 
;
```

## VIEW WITH RECURSIVE

The Teradata format for **RECURSIVE VIEW** keyword is unsupported in DWS. Therefore the keyword is replaced with **VIEW WITH RECURSIVE** keyword as shown in the following figures.
Use the `tdMigrateLOCKOption` configuration parameter to configure migration of query containing the LOCK keyword. If `tdMigrateLOCKOption` is set to false, the tool will skip migration of the query and will log a message.

**Input - VIEW with ACCESS LOCK**

```sql
CREATE OR REPLACE VIEW DP_SVMEDW.S_LCR_909_001_LCRLOAN AS
LOCK TABLE DP_STEDW.S_LCR_909_001_LCRLOAN FOR ACCESS FOR ACCESS
(SELECT RUN_ID, PRODUCT_ID, CURRENCY,
CASHFLOW, ENTITY, LCR,
TIME_BUCKET, MT, Ctl_Id,
File_Id, Business_Date
FROM DP_STEDW.S_LCR_909_001_LCRLOAN );
```

**Output**

```sql
CREATE OR REPLACE VIEW DP_SVMEDW.S_LCR_909_001_LCRLOAN
/* LOCK TABLE DP_STEDW.S_LCR_909_001_LCRLOAN FOR ACCESS */
(SELECT RUN_ID, PRODUCT_ID, CURRENCY,
```
6.12.2.6 Collect Statistics Migration

**COLLECT STAT** is used in Teradata for collecting optimizer statistics, which would be used for query performance. DWS use the **ANALYZE** statement to support **COLLECT STAT**.

See also: **ANALYZE**.

**Input - COLLECT STATISTICS**

```
COLLECT STAT tab1 COLUMN (c1, c2);
```

**Output**

```
ANALYZE tab1 (c1, c2);
```

**Input - COLLECT STATISTICS**

```
COLLECT STATISTICS
  COLUMN (customer_id,customer_name)
  , COLUMN (postal_code)
  , COLUMN (customer_address)
ON customer_t;
```

**Output**

```
ANALYZE customer_t (
  customer_id
  ,customer_name
  ,postal_code
  ,customer_address
)
;
```

**Input - COLLECT STATISTICS with COLUMN**

```
COLLECT STATISTICS
  COLUMN (Order_Date
  ,o_orderID
  )
ON order_t;
```

**Output**

```
ANALYZE order_t (
  Order_Date
  ,Order_ID
)
;
```

**Input - COLLECT STATISTICS with Schema Name**

```
COLLECT STATS COLUMN (
  empno
  ,ename
)
ON ${schemaname}."usrTab1"
;
```
ANALYZE ${schemaname}."usrTab1"
(  
  empno  
  ,ename  
)  
;

6.12.2.7 ACCESS LOCK

ACCESS LOCK is used to allow the user to read the data from a table that may already be locked for the READ or WRITE.

Use the tdMigrateLOCKOption configuration parameter to configure migration of query containing the LOCK keyword. If tdMigrateLOCKOption is set to false, the tool will skip migration of the query and will log a message.

Input - ACCESS LOCK (tdMigrateLOCKOption=True)

LOCKING TABLE tab1 FOR ACCESS
INSERT INTO tab2
SELECT ...
FROM ...
WHERE ...;

Output
/* LOCKING TABLE tab1 FOR ACCESS */
INSERT INTO tab2
SELECT ...
FROM ...
WHERE ...;

6.12.2.8 DBC Columns

DBC.COLUMNS view is a table containing information about table and view columns, or a stored procedure or a macro parameter. Some of the column names are: DatabaseName, TableName, ColumnName, ColumnFormat, ColumnTitle, ColumnType, DefaultValue. In DWS the equivalent table is called information_schema.columns.

This feature requires one time execution of the custom script file MigrationTool/scripts/teradata/db_scripts/mig_fn_get_datatype_short_name.sql.

For more information about the steps to execute the file, refer Structure of Release Package and Prerequisites sections respectively.

The migration tool migrates the following dbc.columns to their corresponding information_schema.columns.

Table 6-19 Migration of dbc.columns to information_schema columns

<table>
<thead>
<tr>
<th>dbc.columns</th>
<th>information_schema.columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColumnName</td>
<td>Column_Name</td>
</tr>
<tr>
<td>ColumnType</td>
<td>mig_fn_get_datatype_short_name (data_Type)</td>
</tr>
<tr>
<td><strong>dbc.columns</strong></td>
<td><strong>information_schema.columns</strong></td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>ColumnLength</td>
<td>character_maximum_length</td>
</tr>
<tr>
<td>DecimalTotalDigits</td>
<td>numeric_precision</td>
</tr>
<tr>
<td>DecimalFractionalDigits</td>
<td>numeric_scale</td>
</tr>
<tr>
<td>datbasename</td>
<td>table_schema</td>
</tr>
<tr>
<td>tablename</td>
<td>table_name</td>
</tr>
<tr>
<td>ColumnId</td>
<td>ordinal_position</td>
</tr>
</tbody>
</table>

The following assumptions are made when migrating dbc.columns:

- The FROM clause will contain only the dbc.columns TABLE NAME.
- COLUMN NAME can be in the form of column_name or schema_name.table_name.column_name.

Migration of dbc.columns is NOT supported for the following cases:

- If the FROM clause has an ALIAS for dbc.columns table name (dbc.columns alias).
- If dbc.columns is combined with other tables (FROM dbc.columns alias1, table1 alias2 OR dbc.columns alias1 join table1 alias2).

**NOTE**

- If the input SELECT statement includes dbc.column COLUMN NAMES directly, then the tool will migrate the column names with the input column name as an ALIAS. For example, the input column name DecimalFractionalDigits is migrated to numeric_scale with an ALIAS DecimalFractionalDigits.

Example: Input
```sql
SEL
    columnid
    ,DecimalFractionalDigits
FROM
    dbc.columns
;
```

Output:
```sql
SELECT
    ordinal_position columnid
    ,numeric_scale DecimalFractionalDigits
FROM
    information_schema.columns
;
```

- For table names and schema names, the migration tool will convert all string values to lowercase. To maintain case-sensitivity, the table/schema names should be within double quotes. In the following input example, "Test" will not be converted to lowercase.

Example: Input
```sql
SELECT
    TableName
FROM
    dbc . columns
WHERE
    dbc.columns.databasename = "Test";
```

**Input:** dbc.columns table with all supported columns
SELECT
   '$AUTO_DB_IP'
,objectdatabasename
,objecttablename
,'$TX_DATE_10'
,'0'
,FirstStepTime
,FirstRespTime
,RowCount
,CAST( RowCount * SUM ( CASE WHEN T2.data_Type = 'CV' THEN T2.character_maximum_length / 3 ELSE T2.character_maximum_length END ) AS DECIMAL( 38, 0 ) )
,'3'
,'BAK_CLR_DATA'
,'2'

FROM TMP_clr_information T1
inner join dbc.columns T2
ON T1.objectdatabasename = T2.DatabaseName
and T1.objecttablename = T2.TableName
WHERE T2.DatabaseName not in ( SELECT
   child
FROM
   dbc.children
WHERE
   child = T2.DatabaseName
   AND( parent = '$FCRM_DB' )
)
group by 1,2,3,4,5,6,7,8,9,11,12,13,14,15;

Output

SELECT
   '$AUTO_DB_IP'
,objectdatabasename
,objecttablename
,'$TX_DATE_10'
,'0'
,FirstStepTime
,FirstRespTime
,RowCount
,CAST( RowCount * SUM ( CASE WHEN mig_fn_get_datatype_short_name ( T2.data_Type ) = 'CV'
THEN T2.character_maximum_length / 3 ELSE T2.character_maximum_length END ) AS DECIMAL( 38, 0 ) )
,'3'
,'BAK_CLR_DATA'
,'2'

FROM
   TMP_clr_information T1
INNER JOIN information_schema.columns T2
ON T1.objectdatabasename = T2.table_schema
AND T1.objecttablename = T2.table_name
WHERE
   NOT EXISTS ( SELECT
      child
      FROM
      dbc.children
      WHERE
      child = T2.table_schema
      AND ( parent = '$FCRM_DB' )
   )
group by 1,2,3,4,5,6,7,8,9,11,12,13,14,15;

Input: dbc.columns table with TABLE NAME

SELECT
   TRIM( ColumnName )
   ,UPPER( dbc.columnsColumnType )
FROM
dbc.columns
WHERE
dbc.columns.databasename = "Test"
ORDER BY
dbc.columns.ColumnId;

Output
SELECT
TRIM(Column_Name)
,UPPER( mig_fn_get_datatype_short_name ( information_schema.columns.data_Type ) )
FROM
information_schema.columns
WHERE
information_schema.columns.table_schema = CASE
 WHEN TRIM("Test") LIKE "%"
 THEN REPLACE( SUBSTR("Test",2,LENGTH("Test")-2),"\"","" )
 ELSE LOWER("Test")
 END
ORDER BY
information_schema.columns.ordinal_position

6.12.2.9 DBC Tables

The Migration Tool migrates dbc.tables to their corresponding
mig_td_ext.vw_td_dbc_tables.

Example: databasename is migrated as mig_td_ext.vw_td_dbc_tables.schemaname.

Input
sel databasename,tablename FROM dbc.tables
WHERE tablekind='T' and trim(databasename) = '<dbname>
AND
( NOT(TRIM(tablename) LIKE ANY ( <excludelist> ) ) )
;

Output
SELECT
mig_td_ext.vw_td_dbc_tables.schemaname
, mig_td_ext.vw_td_dbc_tables.tablename
FROM
mig_td_ext.vw_td_dbc_tables
WHERE
mig_td_ext.vwTd_dbc_tables.tablekind = 'T'
AND TRIM(mig_td_ext.vw_td_dbc_tables.schemaname) = '<dbname>'
AND ( NOT( TRIM(mig_td_ext.vw_td_dbc_tables.tablename) LIKE ANY ( ARRAY[ < excludelist> ] ) ) )

6.12.2.10 DBC Indices

Migration Tool migrates dbc.indices to the corresponding
mig_td_ext.vw_td_dbc_indices.

Example: databasename is migrated as mig_td_ext.vw_td_dbc_tables.schemaname.

Input
sel databasename,tablename FROM dbc.indices
WHERE tablekind='T' and trim(databasename) = '<dbname>
AND

( NOT( (TRIM(tablename) LIKE ANY ( <excludelist> )) ) ) AND indextype IN ( 'Q', 'P' );

Output

```
SELECT mig_td_ext.vw_td_dbc_indices.schemaname, mig_td_ext.vw_td_dbc_indices.tablename
FROM mig_td_ext.vw_td_dbc_indices
WHERE mig_td_ext.vw_td_dbc_indices.tablekind = 'T'
AND TRIM(mig_td_ext.vw_td_dbc_indices.schemaname) = '<dbname>'
AND NOT( TRIM(mig_td_ext.vw_td_dbc_indices.tablename) LIKE ANY ( ARRAY[ < excludelist> ] ) ) )
```

NOTE

In dbc.indices implementation, the query should contain "AND indextype IN ( 'Q', 'P')". If the query does not contain "AND indextype IN ( 'Q', 'P')", then the query is not migrated and Migration Tool logs the following error message:

"Query/statement is not supported as indextype should be mentioned with values 'P' and 'Q'."

6.12.3 DML

6.12.3.1 Overview

This section contains the migration syntax for migrating Teradata DML. The migration syntax decides how the supported keywords/features are migrated.

In Teradata, SQL queries in a file that contain the DML queries for SELECT, INSERT, UPDATE, DELETE and MERGE can be migrated to DWS.

6.12.3.2 INSERT

The Teradata INSERT (short key INS) statement is used to insert records into the table. The Migration Tool supports the INSERT statement.

6.12.3.3 SELECT

Overview

The Teradata SELECT command (short key SEL) is used to specify the table columns from which data is to be retrieved.

ANALYZE is used in DWS for collecting optimizer statistics, which would be used for query performance.

Input: ANALYZE with INSERT

```
INSERT INTO employee(empno, ename) VALUES (1, 'John');
COLLECT STAT on employee;
```
**Output**

INSERT INTO employee(empno, ename)
SELECT 1, 'John';
ANALYZE employee;

**Input: ANALYZE with UPDATE**

UPD employee SET ename = 'Jane'
WHERE ename = 'John';
COLLECT STAT on employee;

**Output**

UPDATE employee SET ename = 'Jane'
WHERE ename = 'John';
ANALYZE employee;

**Input: ANALYZE with DELETE**

DEL FROM employee WHERE ID > 10;
COLLECT STAT on employee;

**Output**

DELETE FROM employee WHERE ID > 10;
ANALYZE employee;

**Order of Clauses**

For Teradata migration of SELECT statements, all the clauses (FROM, WHERE, HAVING and GROUP BY) can be in any order. The tool will not migrate the statement if it contains a QUALIFY as an ALIAS before the FROM clause.

Use the tdMigrateALIAS configuration parameter to configure migration of ALIAS.

**Input: Order of Clauses**

```
SELECT expr1 AS alias1,
     expr2 AS alias2,
     expr3 AS alias3,
     MAX(expr4), ...
FROM tab1 T1 INNER JOIN tab2 T2
   ON T1.c1 = T2.c2 ...
   AND T3.c5 = '010'
   AND ...
WHERE T1.c7 = '000'
   AND ...
HAVING alias1 <> 'IC'
   AND alias2 <> 'IC'
   AND alias3 <> ''
GROUP BY 1, 2, 3;
```

**Output**

```
SELECT expr1 AS alias1,
     expr2 AS alias2,
     expr3 AS alias3,
     MAX(expr4), ...
FROM tab1 T1 INNER JOIN tab2 T2
   ON T1.c1 = T2.c2 ...
   AND T3.c5 = '010'
   AND ...
WHERE T1.c7 = '000'
   AND ...
HAVING expr1 <> 'IC'
GROUP BY 1, 2, 3
```
AND expr2 <> 'IC'
AND expr3 <> ";"

**Input: Order of Clauses**

SELECT
TOP 10 *
GROUP BY
DeptNo
WHERE
empID < 100
FROM
tbl_employee;

**Output**

SELECT *
FROM
tbl_employee
WHERE
empID < 100
GROUP BY
DeptNo LIMIT 10
;

**NOTE**

If the input script contains QUALIFY as an ALIAS before the FROM clause, the Migration Tool will not migrate the statement and copy the input statement verbatim.

**Input: Order of Clauses with QUALIFY as an ALIAS before the FROM clause**

SELECT *
FROM
table1
WHERE
abc = (SELECT
col1 AS qualify
FROM
TABLE
WHERE
col1 = 5
)
;

**Output**

SELECT *
FROM
table1
WHERE
abc = (SELECT
col1 AS qualify
FROM
TABLE
WHERE
col1 = 5
)
;
Extended Group By Clause

The **GROUP BY** clause can be specified if we want the database to group the selected rows based on the value of expr(s). If this clause contains **CUBE**, **ROLLUP** or **GROUPING SETS** extensions, then the database produces super-aggregate groupings in addition to the regular groupings. These features are not available in DWS and these have been addressed using the **UNION ALL** operator.

Use the `extendedGroupByClause` configuration parameter to configure migration of the extended GROUP BY clause.

Use the `tdMigrateALIAS` configuration parameter to configure migration of ALIAS.

**Input: Extended Group By Clause - CUBE**

```sql
SELECT expr1 AS alias1
    , expr2 AS alias2
    , expr3 AS alias3
    , MAX( expr4 ) , ...
FROM tab1 T1 INNER JOIN tab2 T2
    ON T1.c1 = T2.c2 ...
AND T3.c5 = '010'
AND ...
WHERE T1.c7 = '000'
AND ...
HAVING alias1 <> 'IC'
    AND alias2 <> 'IC'
    AND alias3 <> ''
GROUP BY 1, 2, 3 ;
```

**Output**

```sql
SELECT expr1 AS alias1
    , expr2 AS alias2
    , expr3 AS alias3
    , MAX( expr4 ) , ...
FROM tab1 T1 INNER JOIN tab2 T2
    ON T1.c1 = T2.c2 ...
AND T3.c5 = '010'
AND ...
WHERE T1.c7 = '000'
AND ...
GROUP BY 1, 2, 3
HAVING expr1 <> 'IC'
    AND expr2 <> 'IC'
    AND expr3 <> '';
```

**Input: Extended Group By Clause - ROLLUP**

```sql
SELECT d.dname, e.job, MAX(e.sal)
FROM emp e RIGHT OUTER JOIN dept d
    ON e.deptno = d.deptno
WHERE e.job IS NOT NULL
GROUP BY ROLLUP (d.dname, e.job);
```

**Output**

```sql
SELECT dname, job, ColumnAlias1
FROM ( SELECT MAX(e.sal) AS ColumnAlias1, d.dname, e.job
    FROM emp e RIGHT OUTER JOIN dept d
    ON e.deptno = d.deptno
    WHERE e.job IS NOT NULL
GROUP BY d.dname, e.job
UNION ALL
SELECT MAX(e.sal) AS ColumnAlias1, d.dname, NULL AS job
```
FROM emp e RIGHT OUTER JOIN dept d
ON e.deptno = d.deptno
WHERE e.job IS NOT NULL
GROUP BY d.dname
UNION ALL
SELECT MAX( e.sal ) AS ColumnAlias1, NULL AS dname,
NULL AS job
FROM emp e RIGHT OUTER JOIN dept d
ON e.deptno = d.deptno
WHERE e.job IS NOT NULL
);

Input: Extended Group By Clause - GROUPING SETS

SELECT d.dname, e.job, MAX(e.sal)
FROM emp e RIGHT OUTER JOIN dept d
ON e.deptno=d.deptno
WHERE e.job IS NOT NULL
GROUP BY GROUPING SETS(d.dname, e.job);

Output

SELECT dname, job, ColumnAlias1
FROM ( SELECT MAX(e.sal) AS ColumnAlias1
, d.dname, NULL AS job
FROM emp e RIGHT OUTER JOIN dept d
ON e.deptno = d.deptno
WHERE e.job IS NOT NULL
GROUP BY d.dname
UNION ALL
SELECT MAX(e.sal) AS ColumnAlias1
, NULL AS dname, e.job
FROM emp e RIGHT OUTER JOIN dept d
ON e.deptno = d.deptno
WHERE e.job IS NOT NULL
GROUP BY e.job
);

TOP and SAMPLE

The **TOP** and **SAMPLE** clauses of Teradata are migrated to **LIMIT** in DWS.

- **TOP**
  The Migration Tool also supports migration of **TOP** statements with dynamic parameters.

  **NOTE**

  - For **TOP** clauses containing **WITH TIES**, the ORDER BY clause is also required, else the tool will not migrate the statement and copy it as it is.
  - When using TOP with dynamic parameters:
    - The input dynamic parameters should be in the following form:
      ```sql
      TOP :<parameter_name>
      ```
      The following characters are valid for dynamic parameters: a-z, A-Z, 0-9 and "_".

  **Input: SELECT .. TOP**

  ```sql
  SELECT TOP 1 c1, COUNT (*) cnt
  FROM tab1
  GROUP BY c1
  ORDER BY cnt;
  ```

  **Output**

  ```sql
  SELECT c1, COUNT( * ) cnt
  FROM tab1
  ```
GROUP BY c1  
ORDER BY cnt  
LIMIT 1;

**Input: SELECT .. TOP PERCENT**

```sql
SELECT TOP 10 PERCENT c1, c2  
FROM employee  
WHERE ...  
ORDER BY c2 DESC;
```

**Output**

```sql
WITH top_percent AS (  
    SELECT c1, c2  
    FROM employee  
    WHERE ...  
    ORDER BY c2 DESC  
)  
SELECT *  
FROM top_percent  
LIMIT (SELECT CEIL(COUNT( * ) * 10 / 100)  
    FROM top_percent);
```

**Input: SELECT .. TOP with dynamic parameters**

```sql
SELECT  
    TOP :Limit WITH TIES c1  
    ,SUM (c2) sc2  
FROM  
    tab1  
WHERE  
    c3 > 10  
GROUP BY  
    c1  
ORDER BY  
    c1  
;
```

**Output**

```sql
WITH top_ties AS (  
    SELECT  
        c1  
        ,SUM (c2) sc2  
        ,rank (  
            ) OVER( ORDER BY c1 ) AS TOP_RNK  
    FROM  
    tab1  
WHERE  
    c3 > 10  
GROUP BY  
    c1  
)  
SELECT  
    c1  
    ,sc2  
FROM  
    top_ties  
WHERE  
    TOP_RNK <= :Limit  
ORDER BY  
    TOP_RNK  
;
```

**Input: SELECT .. TOP with dynamic parameters and with TIES**

```sql
SELECT  
    TOP :Limit WITH TIES Customer_ID  
FROM  
    Customer_t  
ORDER BY  
    Customer_ID  
;
```

**Output**
WITH top_ties AS (
  SELECT  
    Customer_ID 
    , rank ( ) OVER( order by Customer_id) AS TOP_RNK 
  FROM 
    Customer_t 
) 
SELECT 
  Customer_ID 
FROM 
  top_ties 
WHERE 
  TOP_RNK <= :Limit 
ORDER BY 
  TOP_RNK ;

**Input: SELECT .. TOP PERCENT with dynamic parameters**

```
SELECT  
  TOP :Input_Limit PERCENT WITH TIES c1 
  ,SUM (c2) sc2 
FROM 
  tab1 
GROUP BY 
  c1 
ORDER BY 
  c1 ;
```

**Output**

WITH top_percent_ties AS ( 
  SELECT  
    c1 
    ,SUM (c2) sc2 
    ,rank ( ) OVER( ORDER BY c1 ) AS TOP_RNK 
  FROM 
    tab1 
  GROUP BY 
    c1 
) 
SELECT  
  c1 
  ,sc2 
FROM 
  top_percent_ties 
WHERE 
  TOP_RNK <= ( 
    SELECT  
      CEIL(COUNT( * ) * :Input_Limit / 100) 
    FROM 
      top_percent_ties 
  ) 
ORDER BY 
  TOP_RNK ;

* SAMPLE

**NOTE**

The tool only supports single positive integers in the SAMPLE clause.

**Input: SELECT .. SAMPLE**

```
SELECT c1, c2, c3 
FROM tab1 
WHERE c1 > 1000 
SAMPLE 1; 
```

**Output**
### 6.12.3.4 UPDATE

The **UPDATE** (short key **UPD**) statements are supported and migrated by the tool.

**Input: UPDATE with TABLE ALIAS**

```sql
UPDATE T1
FROM tab1 T1, tab2 T2
SET c1 = T2.c1,
    c2 = T2.c2
WHERE T1.c3 = T2.c3;
```

**Output**

```sql
UPDATE tab1 T1
SET c1 = T2.c1,
    c2 = T2.c2
FROM tab2 T2
WHERE T1.c3 = T2.c3;
```

**Input: UPDATE with TABLE ALIAS using a sub query**

```sql
UPDATE t1
FROM tab1 t1, ( SELECT c1, c2 FROM tab2
    WHERE c2 > 100 ) t2
SET c1 = t2.c1
WHERE t1.c2 = t2.c2;
```

**Output**

```sql
UPDATE tab1 t1
SET c1 = t2.c1
FROM ( SELECT c1, c2 FROM tab2
    WHERE c2 > 100 ) t2
WHERE t1.c2 = t2.c2;
```

**Input: UPDATE with ANALYZE**

```sql
UPD employee SET ename = 'Jane'
    WHERE ename = 'John';
COLLECT STAT on employee;
```

**Output**

```sql
UPDATE employee SET ename = 'Jane'
    WHERE ename = 'John';
ANALYZE employee;
```

### 6.12.3.5 DELETE

**DELETE** (short key **DEL**) is an ANSI-standard SQL syntax operator used to delete existing records from a table. The migration tool supports the Teradata **DELETE**, and the short key **DEL** statements. **DELETE** statements without the **WHERE** clause are migrated to **TRUNCATE** in DWS. Use the **deleteToTruncate** configuration parameter to enable/disable this behavior.

**Input: DELETE**

```sql
DEL FROM tab1
WHERE a =10;
```
### 6.12.3.6 MERGE

**NOTE**

6.5.0 and later versions of the Gauss database support the MERGE function.

**MERGE** is an ANSI-standard SQL syntax operator used to select rows from one or more sources for updating or inserting into a table or view. The conditions to update or insert to the target table or view can be specified.

**Input:** MERGE

```
MERGE INTO tab1 A
using ( SELECT c1, c2, ... FROM tab2 WHERE ... ) AS B
ON A.c1 = B.c1
WHEN MATCHED THEN
  UPDATE SET c2 = c2
  , c3 = c3
WHEN NOT MATCHED THEN
  INSERT VALUES (B.c1, B.c2, B.c3);
```

**Output**

```
WITH B AS ( 
  SELECT
    c1
    , c2
    , ...
  FROM
    tab2
  WHERE
    ...
)
,UPD_REC AS ( 
  UPDATE
    tab1 A
  SET
    c2 = c2
    , c3 = c3
  FROM
    B
  WHERE
    A.c1 = B.c1 returning A. *
)
INSERT INTO
  tab1 SELECT
  B.c1
  , B.c2
  , B.c3
FROM
```

---

**Output**

```
DELETE FROM tab1
WHERE a = 10;
```

**Input:** DELETE without WHERE - Migrated to TRUNCATE if deletetoTruncate=TRUE

```
DELETE FROM ${schemaname} . "tablename" ALL;
```

---

**Output**

```
TRUNCATE
TABLE

${schemaname} . "tablename";
```
WHERE NOT EXISTS (  
  SELECT 1  
  FROM UPD_REC A  
  WHERE A.c1 = B.c1  
)  
;

6.12.3.7 NAMED

NAMED is used in Teradata to assign a temporary name to an expression or column. The NAMED statements for expressions are migrated to AS in DWS. The NAMED statements for column names are retained in the same syntax.

**Input: NAMED Expression migrated to AS**

```sql
SELECT Name, ((Salary + (YrsExp * 200))/12) (NAMED Projection)  
FROM Employee  
WHERE DeptNo = 600 AND Projection < 2500;
```

**Output**

```sql
SELECT Name, ((Salary + (YrsExp * 200))/12) AS Projection  
FROM Employee  
WHERE DeptNo = 600 AND ((Salary + (YrsExp * 200))/12) < 2500;
```

**Input: NAMED AS for Column Name**

```sql
SELECT product_id (NAMED "pid") AS id  
FROM emp where pid=2 or id=2;
```

**Output**

```sql
SELECT product_id AS id  
FROM emp where product_id=2 or product_id=2;
```

**Input: NAMED( ) for Column Name**

```sql
INSERT INTO Neg100 (NAMED,ID,Dept) VALUES ('TEST',1,'IT');
```

**Output**

```sql
INSERT INTO Neg100 (NAMED,ID,Dept) SELECT 'TEST',1, 'IT';
```

**Input: NAMED alias with TITLE alias without AS**

```sql
SELECT dept_name (NAMED alias1) (TITLE alias2)  
FROM employee  
WHERE dept_name like 'Quality';
```

**Output**

```sql
SELECT dept_name AS alias1  
FROM employee  
WHERE dept_name like 'Quality';
```

**Input: NAMED alias with TITLE alias with AS**

The migration tool will skip the NAMED alias and TITLE alias and use only the AS alias.

```sql
SELECT sale_name (Named alias1 ) (Title alias2)  
AS alias3
```
### FROM employee
WHERE sname = 'Stock' OR sname = 'Sales';

**Output**

```sql
SELECT sale_name
   AS alias3
FROM employee
WHERE sname = 'Stock' OR sname = 'Sales';
```

**Input: NAMED with TITLE**

NAMED and TITLE used together, separated by comma within ()

```sql
SELECT customer_id (NAMED cust_id, TITLE 'Customer Id')
FROM Customer_T
WHERE cust_id > 10;
```

**Output**

```sql
SELECT cust_id AS "Customer Id"
FROM (SELECT customer_id AS cust_id
   FROM customer_t
   WHERE cust_id > 10);
```

### 6.12.3.8 ANALYZE

Teradata COLLECT STAT/STATS/STATISTICS used for collecting optimizer statistics, which would be used for query performance. These are migrated to the ANALYZE keyword in DWS.

See also: [Collect Statistics Migration](#).

**Input: ANALYZE with INSERT**

```sql
INSERT INTO employee(empno, ename) VALUES (1, 'John');
COLLECT STAT on employee;
```

**Output**

```sql
INSERT INTO employee( empno, ename)
SELECT 1 , 'John';
ANALYZE employee;
```

**Input: ANALYZE with UPDATE**

```sql
UPD employee SET ename = 'Jane'
   WHERE ename = 'John';
COLLECT STAT on employee;
```

**Output**

```sql
UPDATE employee SET ename = 'Jane'
   WHERE ename = 'John';
ANALYZE employee;
```

**Input: ANALYZE with DELETE**

```sql
DEL FROM employee WHERE ID > 10;
COLLECT STAT on employee;
```

**Output**

```sql
DELETE FROM employee WHERE ID > 10;
ANALYZE employee;
```
### 6.12.3.9 ACTIVITYCOUNT

#### Input

It is a status variable returns the number of rows affected by an SQL DML statement in an embedded SQL.

```
SEL tablename
FROM dbc.tables
WHERE database = 'tera_db'
AND tablename = 'tab1';

IF ACTIVITYCOUNT > 0 THEN .GOTO NXTREPORT;
CREATE MULTISET TABLE tera_db.tab1
  , NO FALLBACK
  , NO BEFORE JOURNAL
  , NO AFTER JOURNAL
  , CHECKSUM = DEFAULT
  ( Tx_Zone_Num CHAR( 4 )
    , Tx_Org_Num VARCHAR( 30 )
  )
PRIMARY INDEX
( Tx_Org_Num
  )
INDEX
( Tx_Teller_Id
  )

.LABEL NXTREPORT
DEL FROM tera_db.tab1;
```

#### Output

```
DECLARE v_verify TEXT;
v_no_data_found NUMBER ( 1 )

BEGIN
  BEGIN
    v_no_data_found := 0;
    SELECT
gd_td_ext.vw_td_dbc_tables.tablename INTO v_verify
    FROM
gd_td_ext.vw_td_dbc_tables
WHERE
gd_td_ext.vw_td_dbc_tables.schemaname = 'tera_db'
AND gd_td_ext.vw_td_dbc_tables.tablename = 'tab1';
  EXCEPTION
  WHEN NO_DATA_FOUND THEN
    v_no_data_found := 1;
  END;
  IF
    v_no_data_found = 1 THEN
      CREATE TABLE tera_db.tab1
        ( Tx_Zone_Num CHAR( 4 )
          , Tx_Org_Num VARCHAR( 30 )
        ) DISTRIBUTE BY HASH ( Tx_Org_Num )
      CREATE INDEX
        ON tera_db.tab1 ( Tx_Teller_Id )
    
```
6.12.3.10 TIMESTAMP

Input - TIMESTAMP with FORMAT

The FORMAT phrase sets the format for a specific TIME or TIMESTAMP column or value. A FORMAT phrase overrides the system format.

```
SELECT 'StartDTTM' as a,
       CURRENT_TIMESTAMP (FORMAT 'HH:MI:SSBMMMBDD,BYYYY');
```

Output

```
SELECT 'StartDTTM' AS a,
       TO_CHAR( CURRENT_TIMESTAMP ,'HH:MI:SS MON DD, YYYY' ) ;
```

6.12.3.11 LOGOFF QUIT

LOGOFF will end the current RDBMS sessions without exiting BTEQ.
QUIT command ends the current Teradata Database sessions and exits BTEQ.

Input

```
SELECT 'StartDTTM' as a
       CURRENT_TIMESTAMP (FORMAT 'HH:MI:SSBMMMBDD,BYYYY');
.LOGOFF;
.QUIT;
```

Output

```
SELECT 'StartDTTM' as a
       CURRENT_TIMESTAMP (FORMAT 'HH:MI:SSBMMMBDD,BYYYY');
.LOGOFF;
.QUIT;
```

**NOTE**

Gauss not supports, so need to comment.

6.12.3.12 COLLECT STATISTICS

Collect statistics using sample percent.

Input

```
COLLECT STATISTICS
USING SAMPLE 5.00 PERCENT
COLUMN ( CDR_TYPE_KEY ) ,
COLUMN ( PARTITION ) ,
COLUMN ( SRC ) ,
COLUMN ( PARTITION,SBSCRPN_KEY )
ON DT_SDM.FCT_OTGO_NTWK_ACTVY_DAILY ;
```

Output

```
SET
default_statistics_target = 5.00 ;
```
6.12.3.13 .IF ERRORCODE and .QUIT

BTEQ statements specified with .IF and .QUIT.

**Input**

```sql
COLLECT STATISTICS
USING SAMPLE 5.00 PERCENT
COLUMN ( CDR_TYPE_KEY ),
COLUMN ( PARTITION ),
COLUMN ( SRC ),
COLUMN ( PARTITION,SBSCRPN_KEY )
ON DT_SDM.FCT_OTGO_NTWK_ACTVY_DAILY ;
```

**Output**

```sql
SET
default_statistics_target = 5.00 ;
ANALYZE DT_SDM.FCT_OTGO_NTWK_ACTVY_DAILY (CDR_TYPE_KEY) ;
ANALYZE DT_SDM.FCT_OTGO_NTWK_ACTVY_DAILY (PARTITION) ;
ANALYZE DT_SDM.FCT_OTGO_NTWK_ACTVY_DAILY (SRC) ;
ANALYZE DT_SDM.FCT_OTGO_NTWK_ACTVY_DAILY (PARTITION,SBSCRPN_KEY) ;
RESET default_statistics_target ;
```

6.12.4 Query Migration Operators

6.12.4.1 Overview

This section contains the migration syntax for migrating Teradata query migration operators. The migration syntax decides how the supported keywords/features are migrated.

6.12.4.2 QUALIFY

In general, the QUALIFY clause is accompanied by analytical functions (window functions) such as DISTINCT, CSUM(), MDIFF(), ROW_NUMBER() and RANK(). This is addressed using sub-query that contains the window functions specified in the QUALIFY clause. Migration tool supports QUALIFY with DISTINCT, MDIFF(), RANK() and ROW_NUMBER(). QUALIFY is a Teradata extension and not an ANSI standard syntax. It is executed after the WHERE and GROUP BY clauses.

**Note**

Migration tool supports column name and/or expressions in the ORDER BY clause only if the column name and/or expression is explicitly included in the SELECT statement as well.

**Input: QUALIFY**

```sql
SELECT
    CUSTOMER_ID,
    CUSTOMER_NAME
FROM
    CUSTOMER_T QUALIFY row_number( ) Over( partition BY CUSTOMER_ID ORDER BY POSTAL_CODE DESC ) = 1;
```
### Output

```sql
SELECT
    CUSTOMER_ID
  , CUSTOMER_NAME
FROM
(  
    SELECT
        CUSTOMER_ID
      , CUSTOMER_NAME
      , row_number() over ( partition BY CUSTOMER_ID ORDER BY POSTAL_CODE DESC ) AS ROW_NUM1
    FROM
        CUSTOMER_T
) Q1
WHERE
    Q1.ROW_NUM1 = 1
;
```

### Input: QUALIFY with multiple ANALYTIC functions and dynamic input parameters

```sql
SELECT
    Customer_Name
  , Postal_Code
FROM
    Customer_t
QUALIFY (  
    Rank ( ) over ( partition BY Customer_Name ORDER BY Customer_City DESC ) IN ( '5' , '1' )
    AND ( row_number() over ( partition BY Customer_Name ORDER BY Customer_City ) = 1 OR row_number() over ( partition BY Customer_Name ORDER BY Customer_City ) = :Customer_value_1 )
)  
;
```

### Output

```sql
SELECT
    Customer_Name
  , Postal_Code
FROM
(  
    SELECT
        Customer_Name
      , Postal_Code
      , Rank ( ) over ( partition BY Customer_Name ORDER BY Customer_City DESC ) AS ROW_NUM1
    FROM
        Customer_t
) Q1
WHERE
    Q1.ROW_NUM1 IN ( '5' , '1' )
    AND ( Q1.ROW_NUM1 = 1 OR Q1.ROW_NUM1 = :Customer_value_1 )
;
```

### Input: QUALIFY with DISTINCT

```sql
SELECT *  
FROM
    Customer_t
WHERE
    Customer_Name NOT IN (  
    SELECT
    )  
;
```
### Output

```sql
SELECT *
FROM Customer_t MigTblAlias1
WHERE NOT EXISTS ( SELECT DISTINCT Customer_Name
FROM Customer_t
WHERE Customer_Name = MigTblAlias1.Customer_Name
GROUP BY 1 )
```

### Input: QUALIFY with MDIFF and RANK

```sql
SELECT material_name
,unit_of_measure * standard_cost AS tot_cost
FROM raw_material_t m LEFT JOIN supplies_t s
ON s.material_id = m.material_id
QUALIFY rank ( ) over( ORDER BY unit_of_measure * standard_cost DESC ) IN '5'
OR mdiff( tot_cost ,3 ,material_name ) IS NULL
```

### Output

```sql
SELECT material_name
,tot_cost
FROM ( SELECT material_name
,unit_of_measure * standard_cost AS tot_cost
FROM raw_material_t m LEFT JOIN supplies_t s
ON s.material_id = m.material_id
QUALIFY rank ( ) over( ORDER BY unit_of_measure * standard_cost DESC ) AS ROW_NUM1
,rank ( ) over( ORDER BY material_name ) AS ROW_NUM2
FROM raw_material_t m LEFT JOIN supplies_t s
ON s.material_id = m.material_id
) Q1
WHERE Q1.ROW_NUM1 = '5'
OR Q1.ROW_NUM2 IS NULL
```

### Input: QUALIFY with ORDER BY having columns that do not exist in the SELECT list

```sql
SELECT Postal_Code
FROM db_pvfc9_std.Customer_t t1
GROUP BY Customer_Name,Postal_Code
QUALIFY ---comments
( Rank ( CHAR(Customer_Address) DESC ) ) = 1
ORDER BY t1.Customer_Name;
```
### Output

```sql
SELECT Postal_Code FROM
  ( SELECT Customer_Name, Postal_Code
    , Rank () over( PARTITION BY Customer_Name, Postal_Code ORDER BY LENGTH(Customer_Address)
      DESC ) AS Rank_col
    FROM db_pvfc9_std.Customer_t t1
  ) Q1
WHERE /*comments*/
  Q1.Rank_col = 1
ORDER BY Q1.Customer_Name;
```

### Input: QUALIFY with COLUMN ALIAS - the corresponding column expression should not be added again in SELECT list.

```sql
SELECT material_name, unit_of_measure * standard_cost as tot_cost,
  RANK() over(order by tot_cost desc) vendor_cnt
FROM raw_material_t m left join supplies_t s
ON s.material_id = m.material_id
QUALIFY vendor_cnt < 5 or MDIFF(tot_cost, 3, material_name) IS NULL;
```

### Output

```sql
SELECT material_name, tot_cost, vendor_cnt
FROM  ( SELECT material_name
    , unit_of_measure * standard_cost AS tot_cost
    , rank () over (ORDER BY tot_cost DESC) vendor_cnt
    FROM raw_material_t m LEFT JOIN supplies_t s
    ON s.material_id = m.material_id
  ) Q1
WHERE  Q1.vendor_cnt < 5 OR Q1.anltfn IS NULL
;  
```

### 6.12.4.3 ALIAS

**ALIAS** is supported by all databases. In Teradata, an **ALIAS** can be referred in SELECT and WHERE clause of the same statement where the alias is defined. Since **ALIAS** is not supported in **SELECT** and **WHERE** clauses in the target, it is replaced by the defined value/expression.

Use the **tdMigrateALIAS** configuration parameter to configure migration of **ALIAS**.

**NOTE**

The comparison operators **LT, LE, GT, GE, EQ, and NE** must not be used as TABLE alias or COLUMN alias.

Tool supports ALIAS name for columns. If the ALIAS name is same as the column name, then it should be specified for that column only and not for other columns in that table. In the following example, there is a conflict between **DATA_DT** column name and the **DATA_DT** alias. This is not supported by the tool.

```sql
SELECT DATA_DT,DATA_INT AS DATA_DT FROM KK WHERE DATA_DT = DATE;
```

### Input: ALIAS

```sql
SELECT expression1 (
  TITLE 'Expression 1'
) AS alias1
,CASE
  WHEN alias1 + Cx >= z
  THEN 1
  ELSE 0
END AS alias2
```
FROM tab1
WHERE
  alias1 = y
;

**Output: tdMigrateALIAS = FALSE**

```sql
SELECT
  expression1 AS alias1,
  CASE
    WHEN alias1 + Cx >= z
    THEN 1
    ELSE 0
  END AS alias2
FROM tab1
WHERE
  alias1 = y
;
```

**Output: tdMigrateALIAS = TRUE**

```sql
SELECT
  expression1 AS alias1,
  CASE
    WHEN expression1 + Cx >= z
    THEN 1
    ELSE 0
  END AS alias2
FROM tab1
WHERE
  expression1 = y
;
```

### 6.12.4.4 FORMAT AND CAST

In Teradata, the `FORMAT` keyword is used for formatting a column/expression. For example, `FORMAT '9(n)'` and `z(n)` are addressed using LPAD with 0 and space (' ') respectively.

Data typing is done using `CAST` or direct data type [like `(expression1)(CHAR(n))`]. This feature is addressed using `CAST`. For more information, see [Type Casting and Formatting](#).

**Input - FORMAT and CAST**

```sql
SELECT
  CAST(TRIM( Agt_Num ) AS DECIMAL( 5 ,0 ) FORMAT '9(5)' )
FROM C03_AGENT_BOND
;

SELECT
  CAST( CAST( Agt_Num AS INT FORMAT 'Z(17)' ) AS CHAR( 5 ) )
FROM C03_AGENT_BOND
;

SELECT
  CHAR( CAST( CAST( CND_VLU AS DECIMAL( 17 ,0 ) FORMAT 'Z(17)' ) AS VARCHAR( 17 ) ) )
FROM C03_AGENT_BOND
;
```

**Output**
```
SELECT LPAD( CAST( TRIM( Agt_Num ) AS DECIMAL( 5 ,0 ) ) ,5 ,0' ) AS Agt_Num FROM C03_AGENT_BOND ;

SELECT CAST( LPAD( CAST( Agt_Num AS INT ) ,17 ,'' ) AS CHAR( 5 ) ) AS Agt_Num FROM C03_AGENT_BOND ;

SELECT LENGTH( CAST( LPAD( CAST( CND_VLU AS DECIMAL( 17 ,0 ) ) ,17 ,'' ) AS VARCHAR( 17 ) ) AS CND_VLU FROM C03_AGENT_BOND ;

Input - FORMAT 'Z(n)9'

SELECT standard_price (FORMAT 'Z(5)9') (CHAR( 6 )) ,max_price (FORMAT 'ZZZZZ9') (CHAR( 6 )) FROM product_t ;

Output

SELECT CAST( TO_CHAR( standard_price ,'999990' ) AS CHAR( 6 ) ) AS standard_price ,CAST( TO_CHAR( max_price ,'999990' ) AS CHAR( 6 ) ) AS max_price FROM product_t ;

Input - FORMAT 'z(m)9.9(n)'

SELECT standard_price (FORMAT 'Z(6)9.9(2)') (CHAR( 6 )) FROM product_t ;

Output

SELECT CAST( TO_CHAR( standard_price ,'9999990.00' ) AS CHAR( 6 ) ) AS standard_price FROM product_t ;

Input - CAST AS INTEGER

SELECT CAST( standard_price AS INTEGER ) FROM product_t ;

Output

SELECT mig_td_ext.mig_fn_castasint (standard_price) FROM product_t ;

Input - CAST AS INTEGER FORMAT
```
SELECT
    CAST( price11 AS INTEGER FORMAT 'Z(4)9' )
    (CHAR(10))
FROM
    product_t
;

Output
SELECT
    CAST( TO_CHAR(mig_td_ext.mig_fn_castasint(price11), '99990') AS CHAR(10)) AS price11
FROM
    product_t
;

NOTE
The following Gauss function is added to convert to integer:

```sql
CREATE OR REPLACE FUNCTION mig_td_ext.mig_fn_castasint
/* This function is used to support "CAST AS INTEGER" of Teradata.
   It should be created in the "mig_td_ext" schema.
*/
(i_param                            TEXT )
RETURN INTEGER
AS
    v_castasint INTEGER;
BEGIN
    v_castasint := CASE WHEN i_param IS NULL
                        THEN NULL         -- if NULL value is provided as input
                        WHEN TRIM(i_param) IS NULL
                        THEN 0                  -- if empty string with one
                        ELSE TRUNC(CAST(i_param AS NUMBER))            -- if any
                        or more spaces is provided
                        THEN NULL           -- if empty string with one
                        ELSE TRUNC(CAST(i_param AS NUMBER))            -- if any
                        numeric value is provided
                        END;
    RETURN v_castasint;
END;
```

6.12.4.5 Short Keys Migration

Table 6-20 lists the supported Teradata short keys and the DWS equivalent syntax.

<table>
<thead>
<tr>
<th>Teradata Short Key</th>
<th>Equivalent DWS Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEL</td>
<td>SELECT</td>
</tr>
<tr>
<td>INS</td>
<td>INSERT</td>
</tr>
<tr>
<td>UPD</td>
<td>UPDATE</td>
</tr>
<tr>
<td>DEL</td>
<td>DELETE</td>
</tr>
<tr>
<td>CT</td>
<td>CREATE TABLE</td>
</tr>
<tr>
<td>CV</td>
<td>CREATE VIEW</td>
</tr>
<tr>
<td>BT</td>
<td>START TRANSACTION</td>
</tr>
<tr>
<td>ET</td>
<td>COMMIT</td>
</tr>
</tbody>
</table>
**Input - BT**

```sql
BT;
--
delete from $(BRTL_DCOR).BRTL_CS_CUST_CID_UID_REL
where DW_Job_Seq = ${v_Group_No};
.if ERRORCODE <> 0 then .quit 12;
--
insert into $(BRTL_DCOR).BRTL_CS_CUST_CID_UID_REL
  (Cust_Id
   ,Cust_UID
   ,DW_Upd_Dt
   ,DW_Upd_Tm
   ,DW_Job_Seq
   ,DW_Etl_Dt)
select
  a.Cust_Id
  ,a.Cust_UID
  ,current_date as Dw_Upd_Dt
  ,current_time(0) as DW_Upd_Tm
  ,cast(${v_Group_No} as byteint) as DW_Job_Seq
  ,cast('${v_Trx_Dt}' as date format 'yyyy-mm-dd') as DW_Etl_Dt
from $(BRTL_VCOR).BRTL_CS_CUST_CID_UID_REL_S a
where a.DW_Snsh_Dt = cast('${v_Trx_Dt}' as date format 'yyyy-mm-dd');
.if ERRORCODE <> 0 then .quit 12;
ET;cd ..
```

**Output**

```sql
BEGIN
--
BEGIN
  delete from $(BRTL_DCOR).BRTL_CS_CUST_CID_UID_REL
  where DW_Job_Seq = ${v_Group_No};
  lv_mig_errorcode = 0;
EXCEPTION
  WHEN OTHERS THEN
  lv_mig_errorcode = -1;
END;
IF lv_mig_errorcode <> 0 THEN
  RAISE EXCEPTION '12';
END IF;
--
BEGIN
  insert into $(BRTL_DCOR).BRTL_CS_CUST_CID_UID_REL
    (Cust_Id
     ,Cust_UID
     ,DW_Upd_Dt
     ,DW_Upd_Tm
     ,DW_Job_Seq
     ,DW_Etl_Dt)
  select
    a.Cust_Id
    ,a.Cust_UID
    ,current_date as Dw_Upd_Dt
    ,current_time(0) as DW_Upd_Tm
    ,cast(${v_Group_No} as byteint) as DW_Job_Seq
    ,cast('${v_Trx_Dt}' as date format 'yyyy-mm-dd') as DW_Etl_Dt
from $(BRTL_VCOR).BRTL_CS_CUST_CID_UID_REL_S a
where a.DW_Snsh_Dt = cast('${v_Trx_Dt}' as date format 'yyyy-mm-dd');
```
6.12.4.6 Object Names Starting With $

This section describes the migration of object names starting with $ (dollar).

The migration behavior for object names starting with $ is explained in the following table. Use the `tdMigrateDollar` configuration parameter to configure the behavior.

See also: IN / NOT IN conversion

<table>
<thead>
<tr>
<th><code>tdMigrateDollar Setting</code></th>
<th><code>Object Name</code></th>
<th><code>Migrated to</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>$V_SQL Static object name starting with $</td>
<td>&quot;$V_SQL&quot;</td>
</tr>
<tr>
<td>true</td>
<td>${V_SQL} Dynamic object name starting with $</td>
<td>${V_SQL} Dynamic object name not supported</td>
</tr>
<tr>
<td>false</td>
<td>$V_SQL Static object name starting with $</td>
<td>$V_SQL No change: Configuration parameter is set to false</td>
</tr>
<tr>
<td>false</td>
<td>${V_SQL} Dynamic object name starting with $</td>
<td>${V_SQL} No change: Configuration parameter is set to false</td>
</tr>
</tbody>
</table>

**NOTE**

Any variable starting with $ is considered as Value. The tool will migrate this by adding ARRAY.

**Input - OBJECT STARTING WITH $**

```sql
CT p11($C1 int,$C2 int);
SELECT $C1 from p11 where $C1 NOT LIKE ANY ($sql1);
```
Output (tdMigrateDollar to TRUE)

```
SELECT
  "$C1"
FROM
  p11
WHERE
  "$C1" NOT LIKE ANY ( ARRAY[ "$sql1" ]
)
```

Output (tdMigrateDollar to FALSE)

```
SELECT
  $C1
FROM
  p11
WHERE
  $C1 NOT LIKE ANY ( ARRAY[ $sql1 ]
)
```

Input - VALUE STARTING WITH $ IN LIKE ALL / LIKE ANY

```
SELECT * FROM T1
WHERE T1.Event_Dt>=ADD_MONTHS(CAST('${OUT_DATE}' AS DATE FORMAT 'YYYYMMDD')+1,
  (-1)*CAST(T7.Tm_Range_Month AS INTEGER))
  AND T1.Event_Dt<=CAST('${OUT_DATE}' AS DATE)
  AND T1.Cntpty_Acct_Name NOT LIKE ANY ( SELECT Tx_Cntpty_Name_Key FROM TEMP_NAME )
  AND T1.Cntpty_Acct_Name NOT LIKE ALL ( SELECT Tx_Cntpty_Name_Key FROM TEMP_NAME )
  AND T1.Cntpty_Acct_Name LIKE ALL ( SELECT Tx_Cntpty_Name_Key FROM TEMP_NAME )
  AND T1.Cntpty_Acct_Name LIKE ANY ( SELECT Tx_Cntpty_Name_Key FROM TEMP_NAME )
  AND T1.Col1 NOT LIKE ANY ($sql1)
  AND T1.Col1 NOT LIKE ALL ($sql1)
  AND T1.Col1 LIKE ANY ($sql1)
  AND T1.Col1 LIKE ALL ($sql1);
```
SELECT S1.col1
FROM $(schema).SALARY ACCT S1
JOIN $(schema).CUST LIST S2
ON S1.Cust_Id = S2.Cust_Id
AND S2.Cust_List = '999'
AND S1.data_dt = -;

Figure 6-6 Input - Dynamic Variable

<table>
<thead>
<tr>
<th>SELECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1.col1</td>
</tr>
<tr>
<td>FROM</td>
</tr>
<tr>
<td>$(schema).SALARY ACCT S1 INNER JOIN $(schema).CUST LIST S2</td>
</tr>
<tr>
<td>ON S1.Cust_Id = S2.Cust_Id</td>
</tr>
<tr>
<td>AND S2.Cust_List = '999'</td>
</tr>
<tr>
<td>AND S1.data_dt = -;</td>
</tr>
</tbody>
</table>

Figure 6-7 Output - Dynamic Variable

<table>
<thead>
<tr>
<th>SELECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1.col1</td>
</tr>
<tr>
<td>FROM</td>
</tr>
<tr>
<td>$(schema).SALARY ACCT S1 INNER JOIN $(schema).CUST LIST S2</td>
</tr>
<tr>
<td>ON S1.Cust_Id = S2.Cust_Id</td>
</tr>
<tr>
<td>AND S2.Cust_List = '999'</td>
</tr>
<tr>
<td>AND S1.data_dt = -</td>
</tr>
</tbody>
</table>

6.12.5 Query Optimization Operators

6.12.5.1 Overview

This section contains the migration syntax for migrating Teradata query optimization operators. The migration syntax decides how the supported keywords/features are migrated.

Use the `inToExists` configuration parameter to configure the migration behavior of `IN/NOT IN` to `EXISTS/NOT EXISTS`. 
By default, this parameter is set to **FALSE**. To enable the 'Query optimization' feature, this parameter must be set to **TRUE**.

### 6.12.5.2 IN and NOT IN Conversion

Teradata queries containing the **IN** / **NOT IN** have been optimized and converted to the **EXISTS** / **NOT EXISTS** operator when being migrated to DWS SQL query. The **IN** / **NOT IN** operator supports single and multiple columns. The Migration Tool will migrate **IN** / **NOT IN** statements only when they exist within the **WHERE** or **ON** clause. The following examples show **IN** to **EXISTS** but are also applicable for **NOT IN** to **NOT EXISTS**

**IN to EXISTS Simple Conversion**

In the following example, the keyword **IN** is provided in the input file. For better optimization, the tool replaces it with the **EXISTS** keyword during the migration process.

**NOTE**

- The migration of **IN** / **NOT IN** statements with nested **IN** / **NOT IN** are not supported and the migrated scripts will be invalid.

```sql
UPDATE tab1
SET b = 123
WHERE b IN ('abc')
AND b IN ( SELECT i
    FROM tab2
    WHERE j NOT IN (SELECT m
                     FROM tab3
                 )
    )
;
```

For migration of **IN**/**NOT IN** statements with sub-queries, comments between the **IN** / **NOT IN** operator and the sub-query (see example) are not supported.

**Example:**

```sql
SELECT *
FROM categories
WHERE category_name
    IN --comment
    ( SELECT category_name
      FROM categories1
    )
ORDER BY category_name;
```

- **Migrating IN / NOT IN statements with object names containing $ and #**
  - Tool will not migrate the query if **TABLE** name or **TABLE ALIAS** starts with $ (dollar).
    ```sql
    SELECT Customer_Name
    FROM Customer_t
    WHERE Customer_ID IN( SELECT Customer_ID FROM Customer_t )
    ;
    ```
  - Tool may migrate the query incorrectly if **COLUMN** name starts with # (hash).
    ```sql
    SELECT Customer_Name
    FROM Customer_t
    WHERE #Customer_ID IN( SELECT #Customer_ID FROM Customer_t )
    ;
    ```

**Input-IN**

```sql
SELECT ...
    FROM tab1 t
    WHERE t.col1 IN (SELECT icol1 FROM tab2 e)
ORDER BY col1
```
### 6.12.6 System Functions and Operators

#### 6.12.6.1 Overview

This section contains the migration syntax for migrating Teradata system functions and operators. The migration syntax decides how the supported keywords/features are migrated.

#### 6.12.6.2 Analytical Functions

Analytical functions are collectively called ordered analytical functions in Teradata, and they provide powerful analytical abilities for data mining, analysis and business intelligence.

### Analytical Functions in ORDER BY

#### Input: Analytic function in ORDER BY clause

```
SELECT customer_id, customer_name, RANK(customer_id, customer_address DESC)
FROM customer_t
WHERE customer_state = ‘CA’
ORDER BY RANK(customer_id, customer_address DESC);
```

#### Output

```
SELECT customer_id, customer_name, RANK() over(order by customer_id, customer_address DESC)
FROM customer_t
```
WHERE customer_state = 'CA'
ORDER BY RANK() over(order by customer_id DESC, customer_address DESC) ;

**Input: Analytic function in GROUP BY clause**

```sql
SELECT customer_city, customer_state, postal_code,
       rank(postal_code), rank() over(partition by customer_state order by postal_code),
       rank() over(order by postal_code)
FROM Customer_T
GROUP BY customer_state
ORDER BY customer_state;
```

**Output**

```sql
SELECT customer_city, customer_state, postal_code,
       rank() over(PARTITION BY customer_state ORDER BY postal_code DESC),
       rank() over(partition by customer_state order by postal_code),
       rank() over(order by postal_code)
FROM Customer_T
ORDER BY customer_state;
```

**Analytical Functions in PARTITION BY**

When the input script contains a numeric value in the PARTITION BY clause, the migrated script retains the numeric value as it is.

**Input: Analytic function in PARTITION BY clause (with numeric value)**

```sql
SELECT Customer_id,
customer_name,
       rank ( ) over( partition BY 1 ORDER BY Customer_id )
FROM Customer_t
GROUP BY 1;
```

**Output**

```sql
SELECT Customer_id,
customer_name,
       rank ( ) over( PARTITION BY Customer_id ORDER BY customer_name DESC )
FROM Customer_t;
```

**6.12.6.3 Window Functions**

Window functions perform calculations across rows of the query result. Migration Tool supports the following Teradata WINDOW functions:

**NOTE**

The Migration Tool supports only single occurrence of WINDOW FUNCTION in QUALIFY clause. Multiple WINDOW functions in a QUALIFY may result in invalid migration.
CSUM

The Cumulative Sum (CSUM) function provides a running or cumulative total for a column's numeric value. It is recommended that ALIAS be used in the QUALIFY statements.

Input - CSUM with GROUP_ID

```sql
INSERT INTO GSIS_SUM.DW_DAT71 (col1 ,PROD_GROUP )
SELECT
    CSUM(1, T1.col1)
 ,T1.PROD_GROUP
FROM tab1 T1
WHERE T1.col1 = 'ABC'
;
```

Output

```sql
INSERT INTO GSIS_SUM.DW_DAT71 (col1 ,PROD_GROUP )
SELECT
    SUM (1) over ( ORDER BY T1.col1 ROWS UNBOUNDED PRECEDING )
 ,T1.PROD_GROUP
FROM tab1 T1
WHERE T1.col1 = 'ABC'
;
```

Input - CSUM with GROUP_ID

```sql
SELECT top 10
    CSUM(1, T1.Test_GROUP)
 ,T1.col1
FROM ${schema}. T1
WHERE T1.Test_GROUP = 'Test_group' group by Test_group order by Test_Group;
```

Output

```sql
SELECT SUM (1) over ( partition BY Test_group ORDER BY T1.Test_GROUP ROWS UNBOUNDED PRECEDING )
 ,T1.col1
FROM ${schema}. T1
WHERE T1.Test_GROUP = 'Test_group'
ORDER BY
    Test_Group LIMIT 10
;
```

Input - CSUM with GROUP BY + QUALIFY

```sql
SELECT c1, c2, c3, CSUM(c4, c3)
FROM tab1
QUALIFY ROW_NUMBER(c4) = 1
GROUP BY 1, 2;
```

Output

```sql
SELECT c1, c2, c3, ColumnAlias1
FROM ( SELECT c1, c2, c3
        , SUM (c4) OVER(PARTITION BY 1 ,2 ORDER BY c3 ROWS UNBOUNDED PRECEDING) AS ColumnAlias1
        , ROW_NUMBER( ) OVER(PARTITION BY 1, 2 ORDER BY c4) AS ROW_NUM1
        FROM tab1
```
MDIFF

The Moving Difference (MDIFF) function provides a moving difference on a column's value, based on a defined number of rows known as the query width. It is recommended that ALIAS be used in the QUALIFY statements.

Input - MDIFF with QUALIFY command

```sql
SELECT DT_A.Acct_ID, DT_A.Trade_Date, DT_A.Stat_PBU_ID,
       CAST( MDIFF( Stat_PBU_ID_3, 1, DT_A.Trade_No ASC ) AS DECIMAL(20,0) ) AS MDIFF_Stat_PBU_ID
FROM Trade_His DT_A
WHERE Trade_Date >= CAST( '20170101' AS DATE )
GROUP BY DT_A.Acct_ID, DT_A.Trade_Date
QUALIFY MDIFF_Stat_PBU_ID <> 0 OR MDIFF_Stat_PBU_ID IS NULL;
```

Output

```sql
SELECT Acct_ID, Trade_Date, Stat_PBU_ID, MDIFF_Stat_PBU_ID
FROM (SELECT DT_A.Acct_ID, DT_A.Trade_Date, DT_A.Stat_PBU_ID,
       CAST( (Stat_PBU_ID_3 - (LAG(Stat_PBU_ID_3, 1, NULL) OVER (PARTITION BY DT_A.Acct_ID, DT_A.Trade_Date ORDER BY DT_A.Trade_No ASC)))  AS MDIFF_Stat_PBU_ID
FROM Trade_His DT_A
WHERE Trade_Date >= CAST( '20170101' AS DATE )
) WHERE MDIFF_Stat_PBU_ID <> 0 OR MDIFF_Stat_PBU_ID IS NULL;
```

RANK

RANK(col1, col2...)

Input: RANK with GROUP BY

```sql
SELECT c1, c2, c3, RANK(c4, c1 DESC, c3) AS Rank1
FROM tab1
WHERE ...
GROUP BY c1;
```

Output

```sql
SELECT c1, c2, c3, RANK() OVER (PARTITION BY c1 ORDER BY c4, c1 DESC ,c3) AS Rank1
FROM tab1
WHERE ...;
```

ROW_NUMBER

ROW_NUMBER(col1, col2...)

Input: ROW NUMBER with GROUP BY + QUALIFY

```sql
SELECT c1, c2, c3, ROW_NUMBER(c4, c3)
FROM tab1
QUALIFY RANK(c4) = 1
GROUP BY 1, 2;
```

Output

```sql
SELECT c1 ,c2 ,c3 ,ColumnAlias1
FROM
```
6.12.6.4 Comparison and List Operators

**NOTE**

The comparison operators LT, LE, GT, GE, EQ, and NE must not be used as TABLE alias or COLUMN alias.

The following comparison and list operators are supported:

`^= and GT`

**Input:** Comparison operations (`^=` and GT)

```sql
SELECT t1.c1, t2.c2
FROM tab1 t1, tab2 t2
WHERE t1.c3 ^= t1.c3
AND t2.c4 GT 100;
```

**Output**

```sql
SELECT t1.c1, t2.c2
FROM tab1 t1, tab2 t2
WHERE t1.c3 <> t1.c3
AND t2.c4 > 100;
```

`EQ and NE`

**Input:** Comparison operations (EQ and NE)

```sql
SELECT t1.c1, t2.c2
FROM tab1 t1 INNER JOIN tab2 t2
ON t1.c2 EQ t2.c2
WHERE t1.c6 NE 1000;
```

**Output**

```sql
SELECT t1.c1, t2.c2
FROM tab1 t1 INNER JOIN tab2 t2
ON t1.c2 = t2.c2
WHERE t1.c6 <> 1000;
```

`LE and GE`

**Input:** Comparison operations (LE and GE)

```sql
SELECT t1.c1, t2.c2
FROM tab1 t1, tab2 t2
WHERE t1.c3 LE 200
AND t2.c4 GE 100;
```

**Output**
**NOT= and LT**

**Input:** Comparison operations (NOT= and LT)

```sql
SELECT t1.c1, t2.c2
FROM tab1 t1, tab2 t2
WHERE t1.c3 NOT= t1.c3
AND t2.c4 LT 100;
```

**Output**

```sql
SELECT t1.c1, t2.c2
FROM tab1 t1, tab2 t2
WHERE t1.c3 <> t1.c3
AND t2.c4 < 100;
```

**IN and NOT IN**

**Also refer IN and NOT IN Conversion for more information.**

**Input:** IN and NOT IN

```sql
SELECT c1, c2
FROM tab1
WHERE c1 IN 'XY';
```

**Output**

```sql
SELECT c1, c2
FROM tab1
WHERE c1 = 'XY';
```

***NOTE***

Even though IN and NOT IN are supported in DWS, in some specific scenarios, DWS does not support these operators.

**IS NOT IN**

**Input:** IS NOT IN

```sql
SELECT c1, c2
FROM tab1
WHERE c1 IS NOT IN (subquery);
```

**Output**

```sql
SELECT c1, c2
FROM tab1
WHERE c1 NOT IN (subquery);
```

**LIKE ALL / NOT LIKE ALL**

**Input:** LIKE ALL / NOT LIKE ALL

```sql
SELECT c1, c2
FROM tab1
WHERE c3 NOT LIKE ALL ('%STR1%', '%STR2%', '%STR3%');
```

**Output**

```sql
SELECT c1, c2
FROM tab1
WHERE c3 NOT LIKE ALL (ARRAY[ '%STR1%', '%STR2%', '%STR3%']);
```
LIKE ANY / NOT LIKE ANY

Input: LIKE ANY / NOT LIKE ANY
SELECT c1, c2
  FROM tab1
  WHERE c3 LIKE ANY ('STR1%', 'STR2%', 'STR3%');

Output
SELECT c1, c2
  FROM tab1
  WHERE c3 LIKE ANY (ARRAY['STR1%', 'STR2%', 'STR3%']);

6.12.6.5 Table Operators

Functions that can be called in the FROM clause of a query from inside the TABLE operator

Input: TABLE operator with RETURNS
SELECT *
  FROM TABLE(sales_retrieve(9005) RETURNS (store INTEGER, item CLOB, quantity BYTEINT)) AS ret;

Output
SELECT *
  FROM sales_retrieve(9005) AS ret (store, item, quantity);

6.12.7 Math Functions

**

Input: **
expr1 ** expr2

Output
expr1 ^ expr2

MOD

Input: MOD
expr1 MOD expr2

Output
expr1 % expr2

NULLIFZERO

Use the tdMigrateNULLIFZERO configuration parameter to configure migration of NULLIFZERO.

Input: NULLIFZERO
SELECT NULLIFZERO(expr1) FROM tab1
WHERE ...;

Output
SELECT NULLIF(expr1, 0) FROM tab1
WHERE ...;
ZEROIFNULL

Use the tdMigrateZEROIFNULL configuration parameter to configure migration of ZEROIFNULL.

Input: ZEROIFNULL
SELECT ZEROIFNULL(expr1) FROM tab1
WHERE ...;

Output
SELECT COALESCE(expr1, 0) FROM tab1
WHERE ...;

6.12.8 String Functions

CHAR function

Input: CHAR
CHAR( expression1 )

Output
LENGTH( expression1 )

CHARACTERS function

Input: CHARACTERS
CHARACTERS( expression1 )

Output
LENGTH( expression1 )

INDEX

Input: INDEX
SELECT INDEX(expr1/string, substring)
FROM tab1
WHERE ...;

Output
SELECT INSTR(expr1/string, substring)
FROM tab1
WHERE ...;

STRREPLACE

Input: STRREPLACE
SELECT STRREPLACE(c2, '.', '')
FROM tab1
WHERE ...;

Output
6.12.9 Date and Time Functions

DATE

Migration tool supports the migration of Teradata DATE FORMAT in SELECT statements, using TO_CHAR to display the date in the source format. This conversion is not done if the date format is an expression (example: Start Dt + 30) or if the WHERE statement contains an expression (Example: WHERE Start Dt > End Dt).

See also: Type Casting to DATE without DATE Keyword

---

### NOTE

- Migration is supported for SELECT statements with and without column alias.
- Date formatting is not supported in the sub levels and in inner queries. It is supported only at the outer query level.
- For date formatting, if a table is created with SCHEMA name, subsequent SELECT statements must also include the schema name. In the following example, the table TEMP_TBL in the SELECT statement will not be migrated and retained as is.

```sql
CREATE TABLE ${SCH}.TEMP_TBL
(C1 INTEGER
,C2 DATE FORMAT 'YYYY-MM-DD')
PRIMARY INDEX(C1,C2);
```

```sql
SELECT ${SCH}.TEMP_TBL.C2 FROM TEMP_TBL where ${SCH}.TEMP_TBL.C2 is not null;
```

---

**Input: DATE FORMAT**

```sql
SELECT CASE
  WHEN SUBSTR( CAST( CAST( SUBSTR( '20180631' ,1 ,6 ) || '01' AS DATE FORMAT 'YYYYMMDD' ) + abc_day - 1 AS FORMAT 'YYYYMMDD' ) ,1 ,6 ) = SUBSTR( '20180631' ,1 ,6 )
    THEN 1
  ELSE 0
END FROM tab1;
```

---

**Output**

```sql
SELECT CASE
  WHEN TO_CHAR( CAST( SUBSTR( '20180631' ,1 ,6 ) || '01' AS DATE ) + abc_day - 1 ,"YYYYMMDD" ) ,1 ,6 ) = SUBSTR("20180631",1 ,6 )
    THEN 1
END FROM tab1;
```
Migration tool supports migration of the date value. If the input DATE is followed by "YYYY-MM-DD" then the date is not changed in the output. The following examples show conversion of DATE to CURRENT_DATE.

**Input: DATE**

```sql
SELECT
t1.c1
t2.c2
FROM
   $schema.tab1 t1
   $schema.tab2 t2
WHERE
t1.c3 ^ = t1.c3
   AND t2.c4 GT DATE
;
```

**Output**

```sql
SELECT
t1.c1
t2.c2
FROM
   "$schema".tab1 t1
   "$schema".tab2 t2
WHERE
t1.c3 <> t1.c3
   AND t2.c4 > CURRENT_DATE
;
```

**Input: DATE with "YYYY-MM-DD"**

```sql
ALTER TABLE
   $abc . tab1 ADD (col_date DATE DEFAULT DATE '2000-01-01')
;
```

**Output**

```sql
ALTER TABLE
   "$abc" . tab1 ADD (col_date DATE DEFAULT DATE '2000-01-01')
;
```

**Input: DATE subtraction**

```sql
SELECT
   CAST( T1.Buyback_Mature_Dt - CAST( '${gsTXDate}' AS DATE FORMAT 'YYYYMMDD' ) AS CHAR( 5 ) )
FROM
   tab1 T1
WHERE
   T1.col1 > 10
;
```

**Output:**

```sql
SELECT
   CAST( EXTRACT( 'DAY' FROM ( T1.Buyback_Mature_Dt - CAST( '${gsTXDate}' AS DATE ) ) ) AS CHAR( 5 ) )
FROM
```
TIMESTAMP

Input: Timestamp

```sql
select CAST('20190811' || ' ' || '01:00:00' AS TIMESTAMP(0) FORMAT 'YYYYMMDDHH:MI:SS') ;
```

Output

```sql
SELECT TO_TIMESTAMP('20190811' || ' ' || '01:00:00', 'YYYYMMDD HH24:MI:SS' ) ;
```

NEXT

Input: NEXT

```sql
SELECT c1, c2
FROM tab1
WHERE NEXT(c3) = CAST('2004-01-04' AS DATE FORMAT 'YYYY-MM-DD');
```

Output

```sql
SELECT c1, c2
FROM tab1
WHERE c3 + 1 = CAST('2004-01-04' AS DATE);
```

6.12.10 Type Casting and Formatting

This section contains the migration syntax for migrating Teradata type casting and formatting syntax. The migration syntax decides how the supported keywords/features are migrated.

In Teradata, the FORMAT keyword is used for formatting a column/expression. FORMAT '9(n)' and 'z(n)' are addressed using LPAD with 0 and space (' ') respectively. Data typing can be done using CAST or direct data type [like (expression1)(CHAR(n))]. This feature is addressed using CAST.

The following type casting and formatting statements are supported by the Migration Tool:

- CHAR
- COLUMNS and COLUMN ALIAS
- Expressions
- INT
- DATE
- DAY to SECOND
- DECIMAL
- Time Interval
- NULL
- Implicit Type Casting Issues
CHAR

**Input - Data type casting for CHAR**

(expression1)(CHAR(n))

**Output**

CAST((expression1) AS CHAR(n))

**COLMUNS and COLUMN ALIAS**

**Input - Type Casting and formatting of a column should have same column name as column alias**

SELECT Product_Line_ID, MAX(Standard_Price)
FROM ( SELECT A.Product_Description, A.Product_Line_ID,
          A.Standard_Price(DECIMAL(18),FORMAT '9(18)')(CHAR(18))
      FROM product_t A
      WHERE Product_Line_ID in (1, 2)
    ) AS tabAls
GROUP BY Product_Line_ID;

**Output**

SELECT Product_Line_ID, MAX( Standard_Price )
FROM ( SELECT A.Product_Description, A.Product_Line_ID,
          CAST( LPAD( CAST(A.Standard_Price AS DECIMAL(18,0)), 18, '0' ) AS CHAR(18) ) AS Standard_Price
      FROM product_t A
      WHERE Product_Line_ID IN(1,2)
    ) AS tabAls
GROUP BY Product_Line_ID;

**Expressions**

**Input - Type Casting and formatting of an expression**

SELECT product_id, standard_price*100.00(DECIMAL(17),FORMAT '9(17)')(CHAR(17)) AS order_amt
FROM db_pvfc9_std.Product_t
WHERE product_line_id is not null;

**Output**

SELECT product_id, CAST(LPAD(CAST(standard_price*100.00 AS DECIMAL(17)), 17, '0') AS CHAR(17)) AS order_amt
FROM db_pvfc9_std.Product_t
WHERE product_line_id is not null;

**INT**

**Input - Data type casting for INT**

SELECT CAST( col1 AS INT ) (FORMAT '9(5)')
FROM table1;

**Output**

SELECT LPAD( CAST( col1 AS INT ),5 ,'0' )
FROM
Input - Data type casting for INT

```
SELECT CAST( col1 AS INT ) ( FORMAT '999999'
) FROM table1
;
```

Output

```
SELECT LPAD( CAST( col1 AS INT ) ,6 ,'0' )
FROM table1
;
```

Input - Data type casting for INT

```
SELECT CAST( expression1 AS INT FORMAT '9(10)' )
FROM table1
;
```

Output

```
SELECT LPAD( CAST( expression1 AS INT ) ,10 ,'0' )
FROM table1
;
```

Input - Data type casting for INT

```
SELECT CAST( expression1 AS INT FORMAT '9999' )
FROM table1
;
```

Output

```
SELECT LPAD( CAST( expression1 AS INT ) ,4 ,'0' )
FROM table1
;
```

**DATE**

In Teradata, when casting DATE from one format to another format, AS FORMAT is used. Migration tool will add TO_CHAR function to retain the specified input format.

See also: [Date and Time Functions](#)

Input - Data type casting without DATE keyword

```
SELECT CAST( '2013-02-12' AS DATE FORMAT 'YYYY/MM/DD' ) AS FORMAT 'DD/MM/YY'
;
```

Output
SELECT TO_CHAR( CAST( '2013-02-12' AS DATE ) ,'DD/MM/YY' ) ;

**DAY to SECOND**

**Input** - Data type casting DAY to SECOND

```sql
SELECT CAST(T1.Draw_Gold_Dt || ' ' || T1.Draw_Gold_Tm AS TIMESTAMP) - CAST(T1.Tx_Dt || ' ' || T1.Tx_Tm AS TIMESTAMP) DAY(4) TO SECOND FROM db_pvfc9_std.draw_tab T1;
```

**Output**

```sql
SELECT
  CAST(( CAST( T1.Draw_Gold_Dt || ' ' || T1.Draw_Gold_Tm AS TIMESTAMP ) - CAST(T1.Tx_Dt || ' ' || T1.Tx_Tm AS TIMESTAMP ) ) AS INTERVAL DAY ( 4 ) TO SECOND )
FROM
  db_pvfc9_std.draw_tab T1;
```

**DECIMAL**

**Input** - Data type casting for DECIMAL

```sql
SELECT
  standard_price ( DECIMAL( 17 ) ,FORMAT '9(17)' ) ( CHAR( 17 ) )
FROM
  db_pvfc9_std.Product_t ;
```

**Output**

```sql
SELECT
  CAST( LPAD( CAST( standard_price AS DECIMAL( 17 ,0 ) ) ,17 ,0' ) AS CHAR( 17 ) )
FROM
  db_pvfc9_std.Product_t ;
```

**Input** - Data type casting for DECIMAL

```sql
SELECT
  standard_price ( DECIMAL( 17 ,0 ) ,FORMAT '9(17)' ) ( VARCHAR( 17 ) )
FROM
  db_pvfc9_std.Product_t ;
```

**Output**

```sql
SELECT
  CAST( LPAD( CAST( standard_price AS DECIMAL( 17 ,0 ) ) ,17 ,0' ) AS VARCHAR( 17 ) )
FROM
  db_pvfc9_std.Product_t ;
```

**Input** - Data type casting for DECIMAL

```sql
SELECT customer_id ( )
```

**Output**

```sql
SELECT
  customer_id ( )
```
DECIMAL(17)
) ( FORMAT '9(17)' ) ( VARCHAR(17)
) FROM db_pvf9_std.Customer_t;

Output

SELECT CAST( LPAD( CAST( customer_id AS DECIMAL(17,0) ),17,'0' ) AS VARCHAR(17) ) FROM db_pvf9_std.Customer_t;

Time Interval

Type casting to time intervals is supported in DDL and DML. It is supported within SELECT and can be used in subqueries of VIEW, MERGE, and INSERT.

Input - Data type casting to TIME Interval

SELECT TIME '06:00:00.00' HOUR TO SECOND;

Output

SELECT TIME '06:00:00.00';

Input - Data type casting to TIME Interval with TOP

SELECT TOP 3 * FROM dwQErrDtl_mc.C03_CORP_AGENT_INSURE WHERE Data_Dt > (SELECT TIME '06:00:00.00' HOUR TO SECOND);

Output

SELECT * FROM dwQErrDtl_mc.C03_CORP_AGENT_INSURE WHERE  Data_Dt > (SELECT TIME '06:00:00.00') limit 3;

NULL

Migration Tool will migrate an expression in the form NULL(data_type) to CAST(NULL AS replacement_data_type).

Input - Data type casting for NULL

NULL(VARCHAR(n))

Output

CAST(NULL AS VARCHAR(n))

Implicit Type Casting Issues

Input - Implicit Type Casting Issues

SELECT Data_Type,Start_Dt,End_Dt
FROM ( SELECT Data_Type,Start_Dt,End_Dt
FROM ( SELECT '101' AS Data_Type,CASE($TX_DATE') AS DATE FORMAT 'YYYYMMDD')-1 AS Start_Dt,CASE($TX_DATE') AS DATE FORMAT 'YYYYMMDD') AS End_Dt
) TT
UNION ALL

SELECT '201' AS Data_Type,CAST('${TX_DATE}' AS DATE FORMAT 'YYYYMMDD')-7 AS Start_Dt,CAST('${TX_DATE}' AS DATE FORMAT 'YYYYMMDD') AS End_Dt
FROM Sys_Calendar.CALENDAR
WHERE calendar_date = CAST('${TX_DATE}' AS DATE FORMAT 'YYYYMMDD')
AND Day_Of_Week = 1
UNION ALL
SELECT Data_Type,Start_Dt,End_Dt
FROM (SELECT '401' AS Data_Type,CAST('${TX_PRIMONTH_END}' AS DATE FORMAT 'YYYYMMDD') AS Start_Dt,CAST('${TX_DATE}' AS DATE FORMAT 'YYYYMMDD') AS End_Dt ) TT
WHERE CAST('${TX_DATE}' AS DATE FORMAT 'YYYYMMDD')=CAST('${TX_MONTH_END}' AS DATE FORMAT 'YYYYMMDD')
UNION ALL
SELECT Data_Type,Start_Dt,End_Dt
FROM (SELECT '501' AS Data_Type,CAST('${TX_PRIQUARTER_END}' AS DATE FORMAT 'YYYYMMDD') AS Start_Dt,CAST('${TX_DATE}' AS DATE FORMAT 'YYYYMMDD') AS End_Dt ) TT
WHERE CAST('${TX_DATE}' AS DATE FORMAT 'YYYYMMDD')=CAST('${TX_QUARTER_END}' AS DATE FORMAT 'YYYYMMDD')
UNION ALL
SELECT Data_Type,Start_Dt,End_Dt
FROM (SELECT '701' AS Data_Type,CAST('${TX_PRIYEAR_END}' AS DATE FORMAT 'YYYYMMDD') AS Start_Dt,CAST('${TX_DATE}' AS DATE FORMAT 'YYYYMMDD') AS End_Dt ) TT
WHERE CAST('${TX_DATE}' AS DATE FORMAT 'YYYYMMDD')=CAST('${TX_YEAR_END}' AS DATE FORMAT 'YYYYMMDD')
)
;

Output
SELECT Data_Type,Start_Dt,End_Dt
FROM (SELECT Data_Type,Start_Dt,End_Dt
FROM (SELECT CAST('101' AS TEXT) AS Data_Type,CAST('${TX_DATE}' AS DATE FORMAT 'YYYYMMDD')-1 AS Start_Dt,CAST('${TX_DATE}' AS DATE FORMAT 'YYYYMMDD') AS End_Dt ) TT
UNION ALL
SELECT CAST('201' AS TEXT) AS Data_Type,CAST('${TX_DATE}' AS DATE FORMAT 'YYYYMMDD')-7 AS Start_Dt,CAST('${TX_DATE}' AS DATE FORMAT 'YYYYMMDD') AS End_Dt
FROM Sys_Calendar.CALENDAR
WHERE calendar_date = CAST('${TX_DATE}' AS DATE FORMAT 'YYYYMMDD')
AND Day_Of_Week = 1
UNION ALL
SELECT Data_Type,Start_Dt,End_Dt
FROM (SELECT CAST('401' AS TEXT) AS Data_Type,CAST('${TX_PRIMONTH_END}' AS DATE FORMAT 'YYYYMMDD') AS Start_Dt,CAST('${TX_DATE}' AS DATE FORMAT 'YYYYMMDD') AS End_Dt ) TT
WHERE CAST('${TX_DATE}' AS DATE FORMAT 'YYYYMMDD')=CAST('${TX_MONTH_END}' AS DATE FORMAT 'YYYYMMDD')
UNION ALL
SELECT Data_Type,Start_Dt,End_Dt
FROM (SELECT CAST('501' AS TEXT) AS Data_Type,CAST('${TX_PRIQUARTER_END}' AS DATE FORMAT 'YYYYMMDD') AS Start_Dt,CAST('${TX_DATE}' AS DATE FORMAT 'YYYYMMDD') AS End_Dt ) TT
WHERE CAST('${TX_DATE}' AS DATE FORMAT 'YYYYMMDD')=CAST('${TX_QUARTER_END}' AS DATE FORMAT 'YYYYMMDD')
UNION ALL
SELECT Data_Type,Start_Dt,End_Dt
FROM (SELECT CAST('701' AS TEXT) AS Data_Type,CAST('${TX_PRIYEAR_END}' AS DATE FORMAT 'YYYYMMDD') AS Start_Dt,CAST('${TX_DATE}' AS DATE FORMAT 'YYYYMMDD') AS End_Dt ) TT
WHERE CAST('${TX_DATE}' AS DATE FORMAT 'YYYYMMDD')=CAST('${TX_YEAR_END}' AS DATE FORMAT 'YYYYMMDD')
)
;
6.13 Oracle Syntax Migration

6.13.1 Overview

The Oracle Migration Syntax lists the Oracle features supported by the Syntax Migration tool, and for each feature provides the Oracle syntax and the equivalent GaussDB 100 syntax. The syntax listed in this section illustrates the internal migration logic used to migrate the Oracle scripts.

It is also a reference for the database migration team and for customer site verification of Oracle scripts migration.

6.13.2 Schema Objects

6.13.2.1 Overview

This section contains the migration syntax for migrating Oracle schema objects. The migration syntax decides how the supported keywords/features are migrated.

6.13.2.2 Tables

CREATE TABLE

The Oracle CREATE TABLE statements are used to create new tables. The target database supports Oracle CREATE TABLES without any migration.

ALTER TABLE

The Oracle ALTER TABLE statement is used to add, rename, modify, or drop/delete columns in a table. The target database supports Oracle ALTER TABLES without any migration.

PRIMARY KEY

The Oracle ALTER TABLE statement adds primary key when the primary key appears in a different file other than the CREATE table statement. After migration, the DDL script must be migrated accordingly.

Input - PRIMARY KEY

CREATE TABLE CTP_ARM_CONFIG
   ( HOSTNAME VARCHAR2(50),
     OPNAME VARCHAR2(50),
     PARAMTYPE VARCHAR2(2),
     PARAMVALUE NUMBER(*,0),
     MODIFYDATE DATE
   ) SEGMENT CREATION DEFERRED
   PCTFREE 10 PCTUSED 0 INITRANS 1 MAXTRANS 255
   NOCOMPRESS LOGGING
   STORAGE( PCTINCREASE 0 )
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE SPMS_DATA ;

ALTER TABLE CTP_ARM_CONFIG ADD CONSTRAINT PKCTP_ARM_CONFIG PRIMARY KEY (HOSTNAME, OPNAME)
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE( PCTINCREASE 0
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE SPMS_DATA  ENABLE;

CREATE TABLE CTP_ARM_CONFIG (HOSTNAME VARCHAR2 (50)
,OPNAME VARCHAR2 (50)
,PARAMTYPE VARCHAR2 (2)
,PARAMVALUE NUMBER (38
,0)
,MODIFYDATE DATE
CONSTRAINT PKCTP_ARM_CONFIG PRIMARY KEY (HOSTNAME
,OPNAME
)
);
/*SEGMENT CREATION DEFERRED*/
/*PCTFREE 10*/
/*PCTUSED 0*/
/*INITRANS 1*/
/*MAXTRANS 255*/
/*NOCOMPRESS*/
/*LOGGING*/
/*STORAGE( BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)*/
/*TABLESPACE SPMS_DATA */
;

Unique

The following ALTER TABLE query contains constraint that shows the error: Cannot create index whose evaluation cannot be enforced to remote nodes.

Implementation is similar to PRIMARY KEY. If PRIMARY KEY / UNIQUE is already present, there is no need to migrate & leave it as it is.

Input

CREATE TABLE GCC_PLAN.T1033 (ROLLOUT_PLAN_LINE_ID NUMBER NOT NULL
,UDF_FIELD_VALUE_ID NUMBER NOT NULL
);

ALTER TABLE GCC_PLAN.T1033 ADD CONSTRAINT UDF_FIELD_VALUE_ID_PK UNIQUE (UDF_FIELD_VALUE_ID) ;

Output

CREATE TABLE GCC_PLAN.T1033
(ROLLOUT_PLAN_LINE_ID NUMBER NOT NULL
,UDF_FIELD_VALUE_ID NUMBER NOT NULL
,CONSTRAINT UDF_FIELD_VALUE_ID_PK UNIQUE (UDF_FIELD_VALUE_ID)
);

NULL Constraint
NULL constraint during local variable declaration is not supported in packages - that is L_CONTRACT_DISTRIBUTE_STATUS
SAD_DISTRIBUTION_HEADERS_T.STATUS%TYPE NULL ;

Input

CREATE OR REPLACE FUNCTION CONTRACT_DISTRIBUTE_STATUS_S2(PI_CONTRACT_NUMBER IN VARCHAR2)
  RETURN VARCHAR2 IS
    L_CONTRACT_DISTRIBUTE_STATUS BAS_SUBTYPE_PKG.STATUS /*NULL*/;
BEGIN
  FOR CUR_CONTRACT IN (SELECT HT.CONTRACT_STATUS
                       FROM SAD_CONTRACTS_V HT
                       WHERE HT.HTH = PI_CONTRACT_NUMBER)
  LOOP
    IF CUR_CONTRACT.CONTRACT_STATUS = 0 THEN
      L_CONTRACT_DISTRIBUTE_STATUS := 'Cancel' ;
    ELSE
      L_CONTRACT_DISTRIBUTE_STATUS := BAS_SUBTYPE_PKG.G_HEADER_WAITING_SPLIT_STATUS;
    END IF;
  END LOOP;
  RETURN L_CONTRACT_DISTRIBUTE_STATUS;
END CONTRACT_DISTRIBUTE_STATUS_S2;
/

Output

CREATE OR REPLACE FUNCTION CONTRACT_DISTRIBUTE_STATUS_S2 ( PI_CONTRACT_NUMBER IN VARCHAR2 )
  RETURN VARCHAR2
PACKAGE
IS
  L_CONTRACT_DISTRIBUTE_STATUS BAS_SUBTYPE_PKG.STATUS /*NULL*/;
BEGIN
  FOR CUR_CONTRACT IN ( SELECT HT.CONTRACT_STATUS
                         FROM SAD_CONTRACTS_V HT
                         WHERE HT.HTH = PI_CONTRACT_NUMBER )
  LOOP
    IF CUR_CONTRACT.CONTRACT_STATUS = 0 THEN
      L_CONTRACT_DISTRIBUTE_STATUS := 'Cancel' ;
    ELSE
      L_CONTRACT_DISTRIBUTE_STATUS := BAS_SUBTYPE_PKG.G_HEADER_WAITING_SPLIT_STATUS;
    END IF;
  END LOOP;
  RETURN L_CONTRACT_DISTRIBUTE_STATUS ;
END ;
/

NO INDEX CREATED

Using Index and storage parameters in alter table should be removed and only the constraint has to be appended while creating table.

Input - PRIMARY KEY

CREATE TABLE CTP_ARM_CONFIG
  ( HOSTNAME VARCHAR2(50),
    OPNAME VARCHAR2(50),
    Data Warehouse Service
    Tool Guide
    6 Migration Tool - SQL Syntax Migration Tool
    Issue 03 (2019-08-08)
    Copyright © Huawei Technologies Co., Ltd.
    447
CREATE TABLE CTP_ARM_CONFIG
(HOSTNAME VARCHAR2 (50),
 OPNAME VARCHAR2 (50),
 PARAMTYPE VARCHAR2 (2),
 PARAMVALUE NUMBER (38, 0),
 MODIFYDATE DATE
) SEGMENT CREATION DEFERRED
PCTFREE 10 PCTUSED 0 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE( PCTINCREASE 0
 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE SPMS_DATA ;
ALTER TABLE CTP_ARM_CONFIG ADD CONSTRAINT PKCTP_ARM_CONFIG PRIMARY KEY
(HOSTNAME, OPNAME)
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE( PCTINCREASE 0
 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE SPMS_DATA ENABLE;

ENABLE

ENABLE in ALTER TABLE MODIFY statement is not supported in GaussDB, hence it is commented in the migrated output.

Input - ENABLE

CREATE TABLE BASE_APPR_FLOW
( FLOW_ID VARCHAR2(5),
 FLOW_NAME VARCHAR2(60),
 FLOW_TYPE VARCHAR2(4),
 FIELD1 NUMBER(5,0),
 FIELD2 NUMBER(5,0),
 NOTES VARCHAR2(40),
 LST_USER_NO VARCHAR2(9),
 LST_USER_STRUCD NUMBER(10,0),
 LST_CHG_DATE VARCHAR2(10),
 FLOW_CONTROL VARCHAR2(200),
 CONT_TYPE VARCHAR2(2)
) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

DISABLE
--- Constraints for Table BASE_APPR_FLOW

ALTER TABLE BASE_APPR_FLOW MODIFY (FLOW_ID NOT NULL ENABLE);
ALTER TABLE BASE_APPR_FLOW MODIFY (FLOW_NAME NOT NULL ENABLE);
ALTER TABLE BASE_APPR_FLOW MODIFY (FLOW_TYPE NOT NULL ENABLE);
ALTER TABLE BASE_APPR_FLOW MODIFY (LST_USER_NO NOT NULL ENABLE);
ALTER TABLE BASE_APPR_FLOW MODIFY (LST_CHG_DATE NOT NULL ENABLE);
ALTER TABLE BASE_APPR_FLOW ADD CONSTRAINT PK_BASE_APPR_FLOW PRIMARY KEY (FLOW_ID)
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE SPMS_DATA ENABLE;

Output

CREATE TABLE
    BASE_APPR_FLOW (  
      FLOW_ID VARCHAR2 (5),
      FLOW_NAME VARCHAR2 (60),
      FLOW_TYPE VARCHAR2 (4),
      FIELD1 NUMBER (5,0),
      FIELD2 NUMBER (5,0),
      NOTES VARCHAR2 (40),
      LST_USER_NO VARCHAR2 (9),
      LST_USER_STRUCD NUMBER (10,0),
      LST_CHG_DATE VARCHAR2 (10),
      FLOW_CONTROL VARCHAR2 (200),
      CONT_TYPE VARCHAR2 (2),
      CONSTRAINT PK_BASE_APPR_FLOW PRIMARY KEY (FLOW_ID)
    ) /*SEGMENT CREATION IMMEDIATE*/
    /*PCTFREE 10*/
    /*PCTUSED 40*/
    /*INITRANS 1*/
    /*MAXTRANS 255*/
    /*NOCOMPRESS*/
    /*LOGGING*/
    /*STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 FREELISTS 1
    FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)*/
    /*TABLESPACE SPMS_DATA */

--- Constraints for Table BASE_APPR_FLOW

ALTER TABLE BASE_APPR_FLOW ALTER FLOW_ID
SET NOT NULL
;

ALTER TABLE BASE_APPR_FLOW ALTER FLOW_NAME
SET NOT NULL
;
ALTER TABLE BASE_APPR_FLOW ALTER FLOW_TYPE SET NOT NULL ;

ALTER TABLE BASE_APPR_FLOW ALTER LST_USER_NO SET NOT NULL ;

ALTER TABLE BASE_APPR_FLOW ALTER LST_CHG_DATE SET NOT NULL ;

PARTITIONS

Maintenance of large tables and indexes can become very time and resource consuming. At the same time, data access performance can reduce drastically for these objects. Partitioning of tables and indexes can benefit the performance and maintenance in several ways.

**Figure 6-8** Partitioning and Sub-partitioning of Tables

Migration Tool supports migration of Range partition.

The tool does not support the following partitions/subpartitions and these are commented in the migrated scripts:

- List partition
- Hash partition
- Range Subpartition
- List Subpartition
- Hash Subpartition

The unsupported partitions/subpartitions may be supported in the future. Configuration parameters have been provided to enable/disable commenting of the unsupported statements. Refer [Configuration Parameters for Oracle Features](#).

PARTITION by HASH

- Hash partitioning is a partitioning technique where a hash algorithm is used to distribute rows evenly across the different partitions (sub-tables). This is typically used where ranges are not appropriate, for example employee ID,
product ID, and so on. Migration Tool does not support PARTITION and SUBPARTITION by HASH and will comment these statements.

**Input - HASH PARTITION**

```sql
CREATE TABLE dept (deptno NUMBER, deptname VARCHAR(32)) PARTITION BY HASH(deptno) PARTITIONS 16;
```

**Output**

```sql
CREATE TABLE dept (deptno NUMBER, deptname VARCHAR(32) /* PARTITION BY HASH(deptno) PARTITIONS 16 */);
```

**Input - HASH PARTITION without partition names**

```sql
CREATE TABLE dept (deptno NUMBER, deptname VARCHAR(32)) PARTITION BY HASH(deptno) PARTITIONS 16;
```

**Output**

```sql
CREATE TABLE dept (deptno NUMBER, deptname VARCHAR(32)) /* PARTITION BY HASH(deptno) PARTITIONS 16 */;
```

**Input - HASH SUBPARTITION**

```sql
CREATE TABLE sales
(prod_id       NUMBER(6)
, cust_id       NUMBER
, time_id       DATE
, channel_id    CHAR(1)
, promo_id      NUMBER(6)
, quantity_sold NUMBER(3)
, amount_sold   NUMBER(10,2)
) PARTITION BY RANGE (time_id) SUBPARTITION BY HASH (cust_id) SUBPARTITIONS 8 STORE IN (ts1, ts2, ts3, ts4)
PARTITION sales_q1_2006 VALUES LESS THAN (TO_DATE('01-APR-2006','dd-MON-yyyy'))
PARTITION sales_q2_2006 VALUES LESS THAN (TO_DATE('01-JUL-2006','dd-MON-yyyy'))
PARTITION sales_q3_2006 VALUES LESS THAN (TO_DATE('01-OCT-2006','dd-MON-yyyy'))
PARTITION sales_q4_2006 VALUES LESS THAN (TO_DATE('01-JAN-2007','dd-MON-yyyy'));
```

**Output**

```sql
CREATE TABLE sales
(prod_id       NUMBER(6)
, cust_id       NUMBER
, time_id       DATE
, channel_id    CHAR(1)
, promo_id      NUMBER(6)
, quantity_sold NUMBER(3)
, amount_sold   NUMBER(10,2)
) PARTITION BY RANGE (time_id) /* SUBPARTITION BY HASH (cust_id) SUBPARTITIONS 8 STORE IN (ts1, ts2, ts3, ts4) */
PARTITION sales_q1_2006 VALUES LESS THAN (TO_DATE('01-APR-2006','dd-MON-yyyy'))
PARTITION sales_q2_2006 VALUES LESS THAN (TO_DATE('01-JUL-2006','dd-MON-yyyy'))
PARTITION sales_q3_2006 VALUES LESS THAN (TO_DATE('01-OCT-2006','dd-MON-yyyy'))
PARTITION sales_q4_2006 VALUES LESS THAN (TO_DATE('01-JAN-2007','dd-MON-yyyy'));
```

- **PARTITION by LIST**

List partitioning is a partitioning technique where you specify a list of discrete values for the partitioning key in the description for each partition. Migration Tool does not support PARTITION and SUBPARTITION by LIST and will comment these statements.

**Input - LIST PARTITION**

```sql
CREATE TABLE sales_by_region (item# INTEGER, qty INTEGER, store_name VARCHAR(30), state_code VARCHAR(2), sale_date DATE) STORAGE(INITIAL 10K NEXT 20K) TABLESPACE tbs5 PARTITION BY LIST (state_code) (PARTITION region_east VALUES ('MA','NY','CT','NH','ME','MD','VA','PA','NJ') STORAGE (INITIAL 8M) TABLESPACE tbs8, PARTITION region_west VALUES ('CA','AZ','NM','OR','WA','UT','NV','CO') NOLOGGING, PARTITION region_south VALUES
```

Data Warehouse Service
Tool Guide

6 Migration Tool - SQL Syntax Migration Tool

Copyright © Huawei Technologies Co., Ltd.
CREATE UNLOGGED TABLE sales_by_region ( item# INTEGER , qty INTEGER , store_name VARCHAR(30) , state_code VARCHAR(2) , sale_date DATE ) TABLESPACE tbs5 /* PARTITION BY LIST(state_code)(PARTITION region_east VALUES('MA','NY','CT','NH','ME','MD','VA','PA','NJ') TABLESPACE tbs8, PARTITION region_west VALUES('CA','AZ','NM','OR','WA','UT','NV','CO') , PARTITION region_south VALUES('TX','KY','TN','LA','MS','AR','AL','GA'), PARTITION region_central VALUES('OH','ND','SD','MO','IL','MI','IA'), PARTITION region_null VALUES(NULL), PARTITION region_unknown VALUES(DEFAULT) ) */;

CREATE TABLE store_master ( Store_id NUMBER , Store_address VARCHAR2 (40) , City VARCHAR2 (30) , State VARCHAR2 (2) , zip VARCHAR2 (10) , manager_id NUMBER ) /*TABLESPACE users*/ STORAGE ( INITIAL 100 k NEXT 100 k PCTINCREASE 0 ) PARTITION BY LIST (city) ( PARTITION south_florida VALUES ( 'MIA', 'ORL' ) /*TABLESPACE users*/ STORAGE ( INITIAL 100 k NEXT 100 k PCTINCREASE 0 ) , PARTITION north_florida VALUES ( 'JAC', 'TAM', 'PEN' ) /*TABLESPACE users*/ STORAGE ( INITIAL 100 k NEXT 100 k PCTINCREASE 0 ) , PARTITION south_georgia VALUES ( 'BRU', 'WAY', 'VAL' ) /*TABLESPACE users*/ STORAGE ( INITIAL 100 k NEXT 100 k PCTINCREASE 0 ) , PARTITION north_georgia VALUES ( 'ATL', 'SAV', NULL ) );

CREATE TABLE tab1_list PARTITION BY LIST (col1) ( partition part1 VALUES ( 1 ) , partition part2 VALUES ( 2, 3, 4 ) , partition part3 VALUES (DEFAULT) ) AS SELECT * FROM tab1;
CREATE TABLE tab1_list
AS
(SELECT *
FROM tab1);

**Input - LIST PARTITION with SUBPARTITIONS**

CREATE TABLE big_t_list PARTITION BY LIST(n10) (partition part1 VALUES (1), partition part2 VALUES (2,3,4), partition part3 VALUES (DEFAULT)) AS SELECT * FROM big_t;

**Output**

CREATE TABLE big_t_list /* PARTITION BY LIST(n10)(partition part1 VALUES(1) ,partition part2 VALUES(2,3,4) ,partition part3 VALUES(DEFAULT)) */ AS (SELECT * FROM big_t);

**Input - LIST PARTITION with SUBPARTITION TEMPLATE**

CREATE TABLE q1_sales_by_region
(deptno NUMBER,
department_name varchar2 (20),
quarterly_sales NUMBER (10,2),
state varchar2 (2))
PARTITION BY LIST (state)
SUBPARTITION BY RANGE (quarterly_sales)
SUBPARTITION TEMPLATE (
SUBPARTITION original VALUES LESS THAN (1001),
SUBPARTITION acquired VALUES LESS THAN (8001),
SUBPARTITION recent VALUES LESS THAN (MAXVALUE)
)
( PARTITION q1_northwest VALUES ('OR', 'WA'),
PARTITION q1_southwest VALUES ('AZ', 'UT', 'NM'),
PARTITION q1_northeast VALUES ('NY', 'VM', 'NJ'),
PARTITION q1_southcentral VALUES ('OK', 'TX'));

**Output**

CREATE TABLE q1_sales_by_region
(deptno NUMBER,
department_name varchar2 (20),
quarterly_sales NUMBER (10,2),
state varchar2 (2));

**PARTITION by RANGE**

Range partitioning is a partitioning technique where ranges of data are stored separately in different sub-tables. Range partitioning is useful when you have distinct ranges of data you want to store together, for example the date field. Migration Tool supports PARTITION by RANGE. It does not support SUBPARTITION for RANGE and will comment these statements.

**Input - RANGE PARTITION (With Storage Parameters)**

CREATE TABLE CCM_TA550002_H
(TSUR_ID VARCHAR2 (10),
ORGAN1_NO VARCHAR2 (10),
ORGAN2_NO VARCHAR2 (10))
partition BY range (ORGAN2_NO);
partition CCM_TA550002_01
VALUES LESS than ('00100') /*
TABLESPACE users */
/*pctfree 10*/
/*initrans 1*/
/*storage(Initial 256 K NEXT 256
K minextents 1 maxextents
unlimited *)*/
,partition CCM_TA550002_02
VALUES LESS than ('00200') /*
TABLESPACE users */
/*pctfree 10*/
/*initrans 1*/
/* storage ( initial 256 K NEXT
256K minextents 1
maxextents unlimited
pctincrease 0 */)
);

Output
CREATE TABLE CCM_TA550002_H
( STRU_ID VARCHAR2 (10)
, ORGAN1_NO VARCHAR2 (10)
, ORGAN2_NO VARCHAR2 (10)
)
partition BY range (ORGAN2_NO)
( partition CCM_TA550002_01 VALUES LESS
than ('00100')
/*TABLESPACE users*/
, partition CCM_TA550002_02 VALUES LESS
than ('00200')
/*TABLESPACE users*/
);

Input - RANGE PARTITION with SUBPARTITIONS
CREATE TABLE composite_rng_list (
cust_id     NUMBER(10),
cust_name   VARCHAR2(25),
cust_state  VARCHAR2(2),
time_id     DATE)
PARTITION BY RANGE(time_id)
/*SUBPARTITION BY LIST (cust_state)
SUBPARTITION Template(
SUBPARTITION west VALUES ('OR', 'WA') TABLESPACE part1,
SUBPARTITION east VALUES ('NY', 'CT') TABLESPACE part2,
SUBPARTITION cent VALUES ('OK', 'TX') TABLESPACE part3)*/
PARTITION per1 VALUES LESS THAN (TO_DATE('01/01/2000','DD/MM/YYYY')),
PARTITION per2 VALUES LESS THAN (TO_DATE('01/01/2005','DD/MM/YYYY')),
PARTITION per3 VALUES LESS THAN (TO_DATE('01/01/2010','DD/MM/YYYY')),
PARTITION future VALUES LESS THAN(MAXVALUE));

Output
CREATE TABLE composite_rng_list (
cust_id     NUMBER(10),
cust_name   VARCHAR2(25),
cust_state  VARCHAR2(2),
time_id     DATE)
PARTITION BY RANGE(time_id)
/*SUBPARTITION BY LIST (cust_state)
SUBPARTITION Template(
SUBPARTITION west VALUES ('OR', 'WA') TABLESPACE part1,
SUBPARTITION east VALUES ('NY', 'CT') TABLESPACE part2,
SUBPARTITION cent VALUES ('OK', 'TX') TABLESPACE part3)*/
PARTITION per1 VALUES LESS THAN (TO_DATE('01/01/2000','DD/MM/YYYY')),
PARTITION per2 VALUES LESS THAN (TO_DATE('01/01/2005','DD/MM/YYYY')),
PARTITION per3 VALUES LESS THAN (TO_DATE('01/01/2010','DD/MM/YYYY')),
PARTITION future VALUES LESS THAN(MAXVALUE));

Input - RANGE PARTITION with SUBPARTITION TEMPLATE
CREATE TABLE composite_rng_rng(
  cust_id NUMBER(10),
  cust_name VARCHAR2(25),
  cust_state VARCHAR2(2),
  time_id DATE)
PARTITION BY RANGE(time_id)
SUBPARTITION BY RANGE (cust_id)
SUBPARTITION TEMPLATE(
  SUBPARTITION original VALUES LESS THAN (1001) TABLESPACE part1,
  SUBPARTITION acquired VALUES LESS THAN (8001) TABLESPACE part2,
  SUBPARTITION recent VALUES LESS THAN (MAXVALUE) TABLESPACE part3)
  PARTITION per1 VALUES LESS THAN (TO_DATE('01/01/2000','DD/MM/YYYY')),
  PARTITION per2 VALUES LESS THAN (TO_DATE('01/01/2005','DD/MM/YYYY')),
  PARTITION per3 VALUES LESS THAN (TO_DATE('01/01/2010','DD/MM/YYYY')),
  PARTITION future VALUES LESS THAN (MAXVALUE));

Output
CREATE TABLE composite_rng_rng(
  cust_id NUMBER(10),
  cust_name VARCHAR2(25),
  cust_state VARCHAR2(2),
  time_id DATE)
  PARTITION BY RANGE(time_id)
  PARTITION per1 VALUES LESS THAN (TO_DATE('01/01/2000','DD/MM/YYYY')),
  PARTITION per2 VALUES LESS THAN (TO_DATE('01/01/2005','DD/MM/YYYY')),
  PARTITION per3 VALUES LESS THAN (TO_DATE('01/01/2010','DD/MM/YYYY')),
  PARTITION future VALUES LESS THAN (MAXVALUE));

PRIMARY KEY/UNIQUE Constraint For Partitioned Table
CREATE TABLE statement with partition by range/hash/List and add constraint gives an error -  
"Invalid PRIMARY KEY/UNIQUE constraint for partitioned table"
NOTE : Columns of PRIMARY KEY/UNIQUE constraint Must contain PARTITION KEY
Scripts : wo_integrate_log_t.sql, wo_change_log_t.sql
Input
CREATE TABLE composite_rng_rng (
  cust_id NUMBER(10),
  cust_name VARCHAR2(25),
  cust_state VARCHAR2(2),
  time_id DATE)
PARTITION BY RANGE(time_id)
SUBPARTITION BY RANGE (cust_id)
SUBPARTITION TEMPLATE(
  SUBPARTITION original VALUES LESS THAN (1001) TABLESPACE part1,
  SUBPARTITION acquired VALUES LESS THAN (8001) TABLESPACE part2,
  SUBPARTITION recent VALUES LESS THAN (MAXVALUE) TABLESPACE part3)
  PARTITION per1 VALUES LESS THAN (TO_DATE('01/01/2000','DD/MM/YYYY')),
  PARTITION per2 VALUES LESS THAN (TO_DATE('01/01/2005','DD/MM/YYYY')),
  PARTITION per3 VALUES LESS THAN (TO_DATE('01/01/2010','DD/MM/YYYY')),
  PARTITION future VALUES LESS THAN (MAXVALUE));
partition SYS_P53873 values less than (TO_DATE(' 2018-11-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
tablespace SDWO_DATA,
partition SYS_P104273 values less than (TO_DATE(' 2018-12-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
tablespace SDWO_DATA,
partition SYS_P105533 values less than (TO_DATE(' 2019-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
tablespace SDWO_DATA,
partition SYS_P108153 values less than (TO_DATE(' 2019-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
tablespace SDWO_DATA,
partition SYS_P127173 values less than (TO_DATE(' 2019-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
tablespace SDWO_DATA,
partition SYS_P130313 values less than (TO_DATE(' 2019-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
tablespace SDWO_DATA;
alter table SD_WO.WO_INTEGRATE_LOG_T
add constraint WO_INTEGRATE_LOG_PK primary key (LOG_ID);
create index SD_WO.WO_INTEGRATE_LOG_N1 on SD_WO.WO_INTEGRATE_LOG_T (BUSINESS_ID);
create index SD_WO.WO_INTEGRATE_LOG_N2 on SD_WO.WO_INTEGRATE_LOG_T (CREATION_DATE, BUSINESS_TYPE);
create index SD_WO.WO_INTEGRATE_LOG_N3 on SD_WO.WO_INTEGRATE_LOG_T (PROJECT_NUMBER, BUSINESS_TYPE);

Output
CREATE TABLE SD_WO.WO_INTEGRATE_LOG_T (LOG_ID NUMBER NOT NULL,
PROJECT_NUMBER VARCHAR2 (40),
MESSAGE_ID VARCHAR2 (100),
BUSINESS_ID VARCHAR2 (100),
BUSINESS_TYPE VARCHAR2 (100),
INTEGRATE_CONTENT CLOB,
OPERATION_RESULT VARCHAR2 (100),
FAILED_MSG VARCHAR2 (4000),
HOST_NAME VARCHAR2 (100) NOT NULL,
CREATED_BY NUMBER NOT NULL,
CREATION_DATE DATE NOT NULL,
LAST_UPDATED_BY NUMBER NOT NULL,
LAST_UPDATE_DATE DATE NOT NULL,
SOURCE_CODE VARCHAR2 (100),
TENANT_ID NUMBER,
CONSTRAINT WO_INTEGRATE_LOG_PK PRIMARY KEY (LOG_ID)
) partition BY range (CREATION_DATE) (partition P2018
VALUES LESS than (TO_DATE( ' 2018-10-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS'/,'NLS_CALENDAR=GREGORIAN*/ ) ) /* tablespace SDWO_DATA */ partition SYS_P53873
VALUES LESS than (TO_DATE( ' 2018-11-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS'/,'NLS_CALENDAR=GREGORIAN*/ ) ) /* tablespace SDWO_DATA */ partition SYS_P104273
VALUES LESS than (TO_DATE( ' 2018-12-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS'/,'NLS_CALENDAR=GREGORIAN*/ ) ) /* tablespace SDWO_DATA */ partition SYS_P105533
VALUES LESS than (TO_DATE( ' 2019-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS'/,'NLS_CALENDAR=GREGORIAN*/ ) ) /* tablespace SDWO_DATA */ partition SYS_P108153
VALUES LESS than (TO_DATE( ' 2019-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS'/,'NLS_CALENDAR=GREGORIAN*/ ) ) /* tablespace SDWO_DATA */ partition SYS_P127173
VALUES LESS than (TO_DATE( ' 2019-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS'/,'NLS_CALENDAR=GREGORIAN*/ ) ) /* tablespace SDWO_DATA */ partition SYS_P130313
CREATE table SD_WO.WO_INTEGRATE_LOG_T
(
    LOG_ID            NUMBER not null,
    PROJECT_NUMBER    VARCHAR2(40),
    MESSAGE_ID        VARCHAR2(100),
    BUSINESS_ID       VARCHAR2(100),
    BUSINESS_TYPE     VARCHAR2(100),
    INTEGRATE_CONTENT CLOB,
    OPERATION_RESULT  VARCHAR2(100),
    FAILED_MSG        VARCHAR2(4000),
    HOST_NAME         VARCHAR2(100) not null,
    CREATED_BY        NUMBER not null,
    CREATION_DATE     DATE not null,
    LAST_UPDATED_BY   NUMBER not null,
    LAST_UPDATE_DATE  DATE not null,
    SOURCE_CODE       VARCHAR2(100),
    TENANT_ID         NUMBER
)
partition by range (CREATION_DATE)
(
    partition P2018 values less than (TO_DATE(' 2018-10-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
tablespace SDWO_DATA,
    partition SYS_P53873 values less than (TO_DATE(' 2018-11-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
tablespace SDWO_DATA,
    partition SYS_P104273 values less than (TO_DATE(' 2018-12-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
tablespace SDWO_DATA,
    partition SYS_P105533 values less than (TO_DATE(' 2019-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
tablespace SDWO_DATA,
    partition SYS_P108153 values less than (TO_DATE(' 2019-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
tablespace SDWO_DATA,
    partition SYS_P127173 values less than (TO_DATE(' 2019-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
tablespace SDWO_DATA,
    partition SYS_P130313 values less than (TO_DATE(' 2019-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
tablespace SDWO_DATA
);
alter table SD_WO.WO_INTEGRATE_LOG_T
    add constraint WO_INTEGRATE_LOG_PK primary key (LOG_ID);
create index SD_WO.WO_INTEGRATE_LOG_N1 on SD_WO.WO_INTEGRATE_LOG_T (BUSINESS_ID);
create index SD_WO.WO_INTEGRATE_LOG_N2 on SD_WO.WO_INTEGRATE_LOG_T (CREATION_DATE, BUSINESS_TYPE);
create index SD_WO.WO_INTEGRATE_LOG_N3 on SD_WO.WO_INTEGRATE_LOG_T (PROJECT_NUMBER, BUSINESS_TYPE);

Output

create table SD_WO.WO_INTEGRATE_LOG_T
(
  LOG_ID            NUMBER not null,
  PROJECT_NUMBER    VARCHAR2(40),
  MESSAGE_ID        VARCHAR2(100),
  BUSINESS_ID       VARCHAR2(100),
  BUSINESS_TYPE     VARCHAR2(100),
  INTEGRATE_CONTENT CLOB,
  OPERATION_RESULT  VARCHAR2(100),
  FAILED_MSG        VARCHAR2(4000),
  HOST_NAME         VARCHAR2(100) not null,
  CREATED_BY        NUMBER not null,
  CREATION_DATE     DATE not null,
  LAST_UPDATED_BY   NUMBER not null,
  LAST_UPDATE_DATE  DATE not null,
  SOURCE_CODE       VARCHAR2(100),
  TENANT_ID         NUMBER
)
partition by range (CREATION_DATE)
(
  partition P2018 values less than (TO_DATE(' 2018-10-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
tablespace SDWO_DATA,
  partition SYS_P53873 values less than (TO_DATE(' 2018-11-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
tablespace SDWO_DATA,
  partition SYS_P104273 values less than (TO_DATE(' 2018-12-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
tablespace SDWO_DATA,
  partition SYS_P105533 values less than (TO_DATE(' 2019-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
tablespace SDWO_DATA,
  partition SYS_P108153 values less than (TO_DATE(' 2019-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
tablespace SDWO_DATA,
  partition SYS_P127173 values less than (TO_DATE(' 2019-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
tablespace SDWO_DATA,
  partition SYS_P130313 values less than (TO_DATE(' 2019-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
tablespace SDWO_DATA
);

alter table SD_WO.WO_INTEGRATE_LOG_T
  add constraint WO_INTEGRATE_LOG_PK primary key (LOG_ID);
create index SD_WO.WO_INTEGRATE_LOG_N1 on SD_WO.WO_INTEGRATE_LOG_T (BUSINESS_ID);
create index SD_WO.WO_INTEGRATE_LOG_N2 on SD_WO.WO_INTEGRATE_LOG_T (CREATION_DATE, BUSINESS_TYPE);
create index SD_WO.WO_INTEGRATE_LOG_N3 on SD_WO.WO_INTEGRATE_LOG_T (PROJECT_NUMBER, BUSINESS_TYPE);

SEGMENT CREATION

SEGMENT CREATION { IMMEDIATE | DEFERRED } is not supported in Gauss, hence it is commented in the migrated output. This is based on the following configuration item: commentStorageParameter=true.

Input - TABLE with SEGMENT CREATION

CREATE TABLE T1
(  MESSAGE_CODE VARCHAR2(50),
  MAIL_TITLE VARCHAR2(1000),
)
STORAGE

Storage parameters including BUFFER_POOL and MAXEXTENTS are not supported in Gauss. Storage parameters are commented when it appears in Tables or Indexes based on the value of the config parameter commmment_storage_parameter.

Input - TABLE with STORAGE

CREATE UNIQUE INDEX PK_BASE_APPR_STEP_DEF ON BASE_APPR_STEP_DEF (FLOW_ID, NODE_ID, STEP_ID)
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE SPMS_DATA ;

CREATE TABLE UFP_MAIL
  ( MAIL_ID NUMBER(*,0),
    MAIL_TITLE VARCHAR2(1000),
    MAIL_BODY VARCHAR2(4000),
    STATUS VARCHAR2(50),
    CREATE_TIME DATE,
    SEND_TIME DATE,
    MAIL_ADDRESS CLOB,
    MAIL_CC CLOB,
    BASE_ID VARCHAR2(20),
    BASE_STATUS VARCHAR2(50),
    BASE_VERIFY VARCHAR2(20),
    BASE_LINK VARCHAR2(4000),
    MAIL_TYPE VARCHAR2(20),
    BLIND_COPY_TO CLOB,
    FILE_NAME VARCHAR2(4000),
    FULL_FILEPATH VARCHAR2(4000)
  ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 0 INITRANS 1 MAXTRANS 255
  NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE Test ;
CREATE TABLE CTP_PROC_LOG
    ( PORC_NAME VARCHAR2(100),
    LOG_TIME VARCHAR2(100),
    LOG_INFO CLOB
    ) SEGMENT CREATION IMMEDIATE
    PCTFREE 10 PCTUSED 0 INITRANS 1 MAXTRANS 255
    NOCOMPRESS LOGGING
    STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
    PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
    BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
    TABLESPACE SPMS_DATA
    LOB (LOG_INFO) STORE AS BASICFILE (
    TABLESPACE SPMS_DATA ENABLE STORAGE IN ROW CHUNK 8192 RETENTION
    NOCACHE LOGGING
    STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
    PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
    BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT))
    ;

If comment_storage_parameter is set TRUE then storage parameters are commented.

STORE keyword for LOB columns is not supported in Gauss, it is commented in the migrated output.

CREATE TABLE CTP_PROC_LOG
    ( PORC_NAME VARCHAR2(100),
    LOG_TIME VARCHAR2(100),
    LOG_INFO CLOB
    ) SEGMENT CREATION IMMEDIATE
    PCTFREE 10 PCTUSED 0 INITRANS 1 MAXTRANS 255
    NOCOMPRESS LOGGING
    STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
    PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
    BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
    TABLESPACE SPMS_DATA
    LOB (LOG_INFO) STORE AS BASICFILE (
    TABLESPACE SPMS_DATA ENABLE STORAGE IN ROW CHUNK 8192 RETENTION
    NOCACHE LOGGING
    STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
    PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
    BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT))
    ;
CREATE TABLE CTP_PROC_LOG (
    PORC_NAME VARCHAR2 (100),
    LOG_TIME VARCHAR2 (100),
    LOG_INFO CLOB
) /*SEGMENT CREATION IMMEDIATE*/
/*PCTFREE 10*/
/*PCTUSED 0*/
/*INITSN 1*/
/*MAXTRANS 255*/
/*NOCOMPRESS*/
/*LOGGING*/
/*STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)*/
/*TABLESPACE SPMS_DATA */
/*LOB (LOG_INFO) STORE AS BASICFILE (TABLESPACE SPMS_DATA ENABLE STORAGE IN ROW CHUNK 8192 RETENTION NOCACHE LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT))*/
;

PCTINCREASE

The storage parameter PCTINCREASE is not supported for all the tables. In addition, all storage parameters (like pctfree, minextents, maxextents) are not allowed for Partition tables.

Input - TABLE with PCTINCREASE

CREATE TABLE tab1 (
    col1 < datatype >,
    col2 < datatype >,
    ...
    , colN < datatype >
) TABLESPACE testts
PCTFREE 10 INITRANS 1 MAXTRANS 255
/* STORAGE (INITIAL 5 M NEXT 5 M MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0 );*/

Output

CREATE TABLE tab1 (
    col1 < datatype >,
    col2 < datatype >,
    ...
    , colN < datatype >
) TABLESPACE testts
PCTFREE 10 INITRANS 1 MAXTRANS 255
/* STORAGE (INITIAL 5 M NEXT 5 M MINEXTENTS 1 MAXEXTENTS UNLIMITED );*/

FOREIGN KEY

A foreign key is a way to enforce referential integrity within an Oracle database. A foreign key means that values in one table must also appear in another table. The referenced table is called the parent table while the table with the foreign key is called the child table. The foreign key in the child table will generally reference a primary key in the parent table. A foreign key can be defined in either a CREATE TABLE statement or an ALTER TABLE statement.

A foreign key constraint must be established with the reference-clause. An inline constraint clause appears as part of the column definition clause or the object
properties clause. An out-of-line constraint appears as part within a relational properties clause or the object properties clause.

If the configuration parameter `foreignKeyHandler` is set to true (default value), then the tool will migrate these statements into commented statements.

Migration tool supports inline and out-of-line foreign key constraints as shown in the following examples.

**Input - Foreign Key with inline constraint in create table**

```
CREATE TABLE orders ( 
    order_no    INT  NOT NULL PRIMARY KEY, 
    order_date   DATE  NOT NULL, 
    cust_id    INT
    [CONSTRAINT fk_orders_cust] 
    REFERENCES customers(cust_id) 
    [ON DELETE SET NULL] 
    [INITIALLY DEFERRED] 
    [ENABLE NOVALIDATE] 
); 
```

**Output**

```
CREATE TABLE orders ( 
    order_no    INT  NOT NULL PRIMARY KEY, 
    order_date   DATE  NOT NULL, 
    cust_id    INT 
    /* 
    [CONSTRAINT fk_orders_cust] 
    REFERENCES customers(cust_id) 
    [ON DELETE SET NULL] 
    [INITIALLY DEFERRED] 
    [ENABLE NOVALIDATE] */ 
); 
```

**Input - Foreign Key with out-of-line constraint in create table**

```
CREATE TABLE customers ( 
    cust_id   INT   NOT NULL, 
    cust_name  VARCHAR(64) NOT NULL, 
    cust_addr  VARCHAR(256), 
    cust_contact_no  VARCHAR(16), 
    PRIMARY KEY (cust_id) 
); 

CREATE TABLE orders ( 
    order_no    INT  NOT NULL, 
    order_date  DATE  NOT NULL, 
    cust_id     INT  NOT NULL, 
    PRIMARY KEY (order_no), 
    CONSTRAINT fk_orders_cust 
    FOREIGN KEY (cust_id) 
    REFERENCES customers(cust_id) 
    ON DELETE CASCADE 
); 
```

**Output**

```
CREATE TABLE customers ( 
    cust_id   INT   NOT NULL, 
    cust_name  VARCHAR(64) NOT NULL, 
    cust_addr  VARCHAR(256), 
    cust_contact_no  VARCHAR(16), 
    PRIMARY KEY (cust_id) 
); 

CREATE TABLE orders ( 
    order_no  INT  NOT NULL, 
    order_date  DATE  NOT NULL, 
    cust_id     INT  NOT NULL, 
    PRIMARY KEY (order_no), 
    CONSTRAINT fk_orders_cust 
    FOREIGN KEY (cust_id) 
    REFERENCES customers(cust_id) 
    ON DELETE CASCADE 
); 
```
order_date  DATE  NOT NULL,
cust_id  INT  NOT NULL,
PRIMARY KEY (order_no) /*,
CONSTRAINT fk_orders_cust
FOREIGN KEY (cust_id)
REFERENCES customers(cust_id)
ON DELETE CASCADE */
);

**number(*,n)**

number(*,n) will be migrated as number(38,n).

**Input – number(*,n)**

```sql
CREATE TABLE NUM1
(STATEMENT_ID VARCHAR2(30),
 PLAN_ID NUMBER,
 TIMESTAMP DATE,
 REMARKS VARCHAR2(4000),
 OPERATION VARCHAR2(30),
 OBJECT_INSTANCE NUMBER(*,0),
 OBJECT_TYPE VARCHAR2(30),
 OPTIMIZER VARCHAR2(255),
 SEARCH_COLUMNS NUMBER,
 ID NUMBER(*,0),
 PARENT_ID NUMBER(*,0),
 DEPTH NUMBER(*,0),
 POSITION NUMBER(*,0),
 COST NUMBER(*,0),
 CARDINALITY NUMBER(*,0),
 BYTES NUMBER(*,0),
 OTHER_TAG VARCHAR2(255),
 PARTITION_START VARCHAR2(255),
 PARTITION_STOP VARCHAR2(255),
 PARTITION_ID NUMBER(*,0),
 OTHER_LONG,
 DISTRIBUTION VARCHAR2(30),
 CPU_COST NUMBER(*,0),
 IO_COST NUMBER(*,0),
 TEMP_SPACE NUMBER(*,0),
 FILTER_PREDICATES VARCHAR2(4000),
 PROJECTION VARCHAR2(4000),
 TIME NUMBER(*,0),
 OTHER_XML CLOB
)
```

**Output**

```sql
CREATE TABLE NUM1
(STATEMENT_ID VARCHAR2 (30),
 PLAN_ID NUMBER,
 TIMESTAMP DATE,
 REMARKS VARCHAR2 (4000),
 OPERATION VARCHAR2 (30),
 OBJECT_INSTANCE NUMBER (38,0),
 OBJECT_TYPE VARCHAR2 (30),
 OPTIMIZER VARCHAR2 (255),
 SEARCH_COLUMNS NUMBER,
 ID NUMBER (38,0),
 PARENT_ID NUMBER (38,0),
 DEPTH NUMBER (38,0),
 POSITION NUMBER (38,0),
 COST NUMBER (38,0),
 CARDINALITY NUMBER (38,0),
 BYTES NUMBER (38,0),
 OTHER_TAG VARCHAR2 (255),
 PARTITION_START VARCHAR2 (255),
 PARTITION_STOP VARCHAR2 (255),
 PARTITION_ID NUMBER (38,0),
 OTHER_LONG,
 DISTRIBUTION VARCHAR2 (30),
 CPU_COST NUMBER (38,0),
 IO_COST NUMBER (38,0),
 TEMP_SPACE NUMBER (38,0),
 FILTER_PREDICATES VARCHAR2 (4000),
 PROJECTION VARCHAR2 (4000),
 TIME NUMBER (38,0),
 OTHER_XML CLOB
)
```
LONG Data Type

Columns defined as LONG can store variable-length character data containing up to 2 gigabytes of information. The tool supports LONG data types in TABLE structure and PL/SQL.

### Input - LONG data-type in table structure

```
CREATE TABLE project ( proj_cd INT,
   , proj_name VARCHAR2(32)
   , dept_no INT
   , proj_det LONG );
```

### Output

```
CREATE TABLE project ( proj_cd INT,
   , proj_name VARCHAR2(32)
   , dept_no INT
   , proj_det TEXT );
```

### Input - LONG data-type in PL/SQL

```
CREATE OR REPLACE FUNCTION fn_proj_det
   ( i_proj_cd INT )
RETURN LONG
IS
   v_proj_det LONG;
BEGIN
   SELECT proj_det
   INTO v_proj_det
   FROM project
   WHERE proj_cd = i_proj_cd;
   RETURN v_proj_det;
END;
/
```

### Output

```
CREATE OR REPLACE FUNCTION fn_proj_det
   ( i_proj_cd INT )
RETURN TEXT
IS
   v_proj_det TEXT;
BEGIN
   SELECT proj_det
   INTO v_proj_det
   FROM project
   WHERE proj_cd = i_proj_cd;
   RETURN v_proj_det;
END;
/
```
SUPPLEMENTAL LOG DATA

Additional columns can be logged in the redo log files. The process of logging these additional columns is called supplemental logging. This feature is supported by Oracle and this is not available in Gauss.

Input

```sql
CREATE TABLE sad.fnd_lookup_values_t
(
  lookup_code_id NUMBER NOT NULL /* ENABLE */
, lookup_code VARCHAR2 (40) NOT NULL /* ENABLE */
, meaning VARCHAR2 (100)
, other_meaning VARCHAR2 (100)
, order_by_no NUMBER
, start_time DATE DEFAULT SYSDATE NOT NULL /* ENABLE */
, end_time DATE
, enable_flag CHAR(1) DEFAULT 'Y' NOT NULL /* ENABLE */
, disable_date DATE
, created_by NUMBER (15,0) NOT NULL /* ENABLE */
, creation_date DATE NOT NULL /* ENABLE */
, last_updated_by NUMBER (15,0) NOT NULL /* ENABLE */
, last_update_date DATE NOT NULL /* ENABLE */
, last_update_login NUMBER (15,0)
, description VARCHAR2 (500)
, lookup_type_id NUMBER NOT NULL/* ENABLE */
, attribute4 VARCHAR2 (250)
, supplemental log data (ALL) COLUMNS
);
```

Output

```sql
CREATE TABLE sad.fnd_lookup_values_t
(
  lookup_code_id NUMBER NOT NULL /* ENABLE */
, lookup_code VARCHAR2 (40) NOT NULL /* ENABLE */
, meaning VARCHAR2 (100)
, other_meaning VARCHAR2 (100)
, order_by_no NUMBER
, start_time DATE DEFAULT SYSDATE NOT NULL /* ENABLE */
, end_time DATE
, enable_flag CHAR(1) DEFAULT 'Y' NOT NULL /* ENABLE */
, disable_date DATE
, created_by NUMBER (15,0) NOT NULL /* ENABLE */
, creation_date DATE NOT NULL /* ENABLE */
, last_updated_by NUMBER (15,0) NOT NULL /* ENABLE */
, last_update_date DATE NOT NULL /* ENABLE */
, last_update_login NUMBER (15,0)
, description VARCHAR2 (500)
, lookup_type_id NUMBER NOT NULL/* ENABLE */
, attribute4 VARCHAR2 (250)
/* ,supplemental log data (ALL) COLUMNS */
);
```

**NOTE**

Supplemental log data not supported by Gauss, need to comment.

"SUPPLEMENTAL LOG DATA" in Table creation is not supported. So, Supplemental Log Data should be commented.

Input

```sql
CREATE TABLE SAD.FND_DATA_CHANGE_LOGS_T
(    LOGID NUMBER,
    TABLE_NAME VARCHAR2(40) NOT NULL ENABLE,
    TABLE_KEY_COLUMNS VARCHAR2(200),
    TABLE_KEY_VALUES VARCHAR2(200),
);
COLUMN_NAME VARCHAR2(40) NOT NULL ENABLE,
COLUMN_CHANGE_FROM_VALUE VARCHAR2(200),
COLUMN_CHANGE_TO_VALUE VARCHAR2(200),
DESCRIPTION VARCHAR2(500),
SUPPLEMENTAL LOG DATA (ALL) COLUMNS
);

Output

CREATE TABLE sad.fnd_data_change_logs_t
(
    logid                    NUMBER,
    ,table_name               VARCHAR2 (40) NOT NULL /* ENABLE */
    ,table_key_columns        VARCHAR2 (200)
    ,table_key_values         VARCHAR2 (200)
    ,column_name              VARCHAR2 (40) NOT NULL /* ENABLE */
    ,column_change_from_value VARCHAR2 (200)
    ,column_change_to_value   VARCHAR2 (200)
    ,description              VARCHAR2 (500)
    /*, SUPPLEMENTAL LOG DATA (ALL) COLUMNS*/
)

Support of COMPRESS Phrase

Input – COMPRESS Phrase

CREATE TABLE test_tab (id            NUMBER(10)    NOT NULL,
description   VARCHAR2(100) NOT NULL,
created_date  DATE          NOT NULL,
created_by    VARCHAR2(50)  NOT NULL,
updated_date  DATE,
updated_by    VARCHAR2(50)
)

NOCOMPRESS
PARTITION BY RANGE (created_date) (PARTITION test_tab_q1 VALUES LESS THAN (TO_DATE('01/04/2003', 'DD/MM/YYYY')) COMPRESS,
PARTITION test_tab_q2 VALUES LESS THAN (MAXVALUE)
);

Output

CREATE TABLE test_tab (id NUMBER (10) NOT NULL
description VARCHAR2 (100) NOT NULL,
created_date DATE NOT NULL,
created_by VARCHAR2 (50) NOT NULL,
updated_date DATE,
updated_by VARCHAR2 (50)
) /*NOCOMPRESS*/
PARTITION BY RANGE (created_date) (PARTITION test_tab_q1 VALUES LESS THAN (TO_DATE('01/04/2003', 'DD/MM/YYYY')) /*COMPRESS*/,
PARTITION test_tab_q2 VALUES LESS THAN (MAXVALUE)
);

Support Bitmap index

There is a config parameter that is introduced for this feature named BitmapIndexSupport default value is comment then the sample input and output is as follows:
**Input – Bitmap index**

CREATE BITMAP INDEX
emp_bitmap_idx
ON index_demo (gender);

**Output**

/*CREATE BITMAP INDEX emp_bitmap_idx ON index_demo (gender);*/

However if the configuration parameter is set to **BTREE** then the output is as follows:

**Output**

CREATE

/*bitmap*/
INDEX emp_bitmap_idx
ON index_demo
USING btree (gender) ;

---

**LONG RAW**

"Data type LONG RAW" in Table creation not supported. Long Raw data type is replaced with Bytea.

**Input**

CREATE TABLE SAD.HW_WORKFLOWDEFS
( ID NUMBER(*,0),
  WF_NAME VARCHAR2(200),
  WF_DEFINITION LONG RAW,
  WF_VERSION NUMBER(*,0),
  WF_PUBLISH CHAR(1),
  WF_MAINFLOW CHAR(1),
  WF_APP_NAME VARCHAR2(20),
  CREATED_BY NUMBER,
  CREATION_DATE DATE,
  LAST_UPDATED_BY NUMBER,
  LAST_UPDATE_DATE DATE,
  WFDesc VARCHAR2(2000) );

**Output**

CREATE TABLE sad.hw_workflowdefs
( id                    NUMBER (38, 0),
  wf_name          VARCHAR2 (200),
  wf_definition  BYTEA,
  wf_version       NUMBER (38, 0),
  wf_publish       CHAR(1),
  wf_mainflow      CHAR(1),
  wf_app_name  VARCHAR2 (20),
  created_by       NUMBER,
  creation_date    DATE,
  last_updated_by  NUMBER,
  last_update_date DATE,
  wfdesc               VARCHAR2 (2000) );

**SYS_GUID**

SYS_GUID is a built-in function which returns the Global Unique Identifier (GUID) for a row in a table. It accepts no arguments and returns a RAW value of 16 bytes.

**Input**
CREATE TABLE sad.fnd_data_change_logs_t
(
  logid                    NUMBER,
  table_name               VARCHAR2 (40) NOT NULL /* ENABLE */
 , table_key_columns        VARCHAR2 (200),
  table_key_values         VARCHAR2 (200),
  column_name              VARCHAR2 (40) NOT NULL /* ENABLE */
 , column_change_from_value VARCHAR2 (200),
  column_change_to_value   VARCHAR2 (200),
  organization_id          NUMBER,
  created_by               NUMBER (15, 0) NOT NULL /* ENABLE */
 , creation_date            DATE NOT NULL /* ENABLE */
 , last_updated_by          NUMBER (15, 0) NOT NULL /* ENABLE */
 , last_update_date         DATE NOT NULL /* ENABLE */
 , last_update_login        NUMBER (15, 0) DEFAULT 0 NOT NULL /* ENABLE */
 , description              VARCHAR2 (500),
  sys_id                   VARCHAR2 (32) DEFAULT MIG_ORA_EXT.Sys_guid( )
 /*, SUPPLEMENTAL LOG DATA (ALL) COLUMNS*/
);

Output

CREATE TABLE sad.fnd_data_change_logs_t
(
  logid                    NUMBER,
  table_name               VARCHAR2 (40) NOT NULL /* ENABLE */
 , table_key_columns        VARCHAR2 (200),
  table_key_values         VARCHAR2 (200),
  column_name              VARCHAR2 (40) NOT NULL /* ENABLE */
 , column_change_from_value VARCHAR2 (200),
  column_change_to_value   VARCHAR2 (200),
  organization_id          NUMBER,
  created_by               NUMBER (15, 0) NOT NULL /* ENABLE */
 , creation_date            DATE NOT NULL /* ENABLE */
 , last_updated_by          NUMBER (15, 0) NOT NULL /* ENABLE */
 , last_update_date         DATE NOT NULL /* ENABLE */
 , last_update_login        NUMBER (15, 0) DEFAULT 0 NOT NULL /* ENABLE */
 , description              VARCHAR2 (500),
  sys_id                   VARCHAR2 (32) DEFAULT MIG_ORA_EXT.Sys_guid( )
 /*, SUPPLEMENTAL LOG DATA (ALL) COLUMNS*/
);

6.13.2.3 Temporary Tables

DWS do not support **GLOBAL TEMPORARY TABLE**. Hence, this has been handled using **LOCAL TEMPORARY TABLE**.

Similarly, **ON COMMIT DELETE ROWS** is also not supported. This has been handled using **ON COMMIT PRESERVE ROWS**.

**Figure 6-9** Input - Temporary Tables

```
CREATE
GLOBAL TEMPORARY TABLE
schema1.temp_tbl1 (col1 VARCHAR2 (400), col2 DATE NOT NULL)
ON COMMIT DELETE ROWS
;
```
6.13.2.4 Global Temporary Tables

Global temp table is modified to local temp table.

**Input - GLOBAL TEMPORARY TABLE**

```sql
CREATE GLOBAL TEMPORARY TABLE
"Pack1"."GLOBAL_TEMP_TABLE"
(
"ID" VARCHAR2(8)
) ON COMMIT DELETE ROWS ;
```

**Output**

```sql
CREATE
LOCAL TEMPORARY TABLE
"Pack1_GLOBAL_TEMP_TABLE" ( 
"ID" VARCHAR2 (8)
)
ON COMMIT PRESERVE ROWS ;
```

6.13.2.5 Indexes

In DWS, during index creation, schema name cannot be specified along with index name. The index will be automatically created in the schema where the index table is created.

**Input - Indexes**

![Image of input index code](image1)

**Output - Indexes**

![Image of output index code](image2)

**Input - Function based INDEX by using CASE**

Function based INDEX is an index that is created on the results of a function or expression on top of a column.
Function based INDEX is an index that is created on the results of a function or expression on top of a column.

Output

```sql
CREATE UNIQUE index GCC_RSRC_ASSIGN_U1
ON GCC_PLAN.GCC_RSRC_ASSIGN_T (
  (CASE
    WHEN( ENABLE_FLAG = 'Y' AND ASSIGN_TYPE = '13' AND WORK_ORDER_ID IS NOT NULL )
    THEN WORK_ORDER_ID
    ELSE NULL
  END)
);
```

**NOTE**

Need to put the expression or function inside ()

**Input - Function based INDEX by using DECODE**

Function based INDEX is an index that is created on the results of a function or expression on top of a column.

CREATE UNIQUE index GCC_PLAN_N2
ON GCC_PLAN.GCC_PLAN_T (
  DECODE (
    ENABLE_FLAG,
    'Y',
    BUSINESS_ID,
    NULL
  )
);

Output

```sql
CREATE UNIQUE index GCC_PLAN_N2
ON GCC_PLAN.GCC_PLAN_T (
  (DECODE (
    ENABLE_FLAG,
    'Y',
    BUSINESS_ID,
    NULL
  ))
);
```

**NOTE**

Need to put the expression or function inside ()

**ORA_HASH**

ORA_HASH is a function that computes a hash value for a given expression or column. If this function is specified on the column(s) of create index, this function will be removed.

**Input**

```sql
create index SD_WO.WO_WORK_ORDER_T_N3 on SD_WO.WO_WORK_ORDER_T (PROJECT_NUMBER,
ORA_HASH(WORK_ORDER_NAME));
```

**Output**

```sql
CREATE index WO_WORK_ORDER_T_N3
ON SD_WO.WO_WORK_ORDER_T (PROJECT_NUMBER,
  ORA_HASH( WORK_ORDER_NAME )
);
**Decode**

CREATE INDEX statement DECODE function used as a part of column that is showing error - syntax error at or near 'DECODE ('

Script - gcc_plan_t.sql

**Input**

```sql
create unique index GCC_PLAN.GCC_PLAN_N2 on GCC_PLAN.GCC_PLAN_T
(DECODE(ENABLE_FLAG,'Y',BUSINESS_ID,NULL));
```

**Output**

```sql
CREATE UNIQUE index GCC_PLAN_N2
ON GCC_PLAN.GCC_PLAN_T ( 
DECODE ( 
ENABLE_FLAG , 'Y' , 
BUSINESS_ID , 
NULL )
); 
```

**Input**

Expected Output -

```sql
CREATE
UNIQUE index GCC_PLAN_N2
ON GCC_PLAN.GCC_PLAN_T ( 
(DECODE ( 
ENABLE_FLAG , 'Y' , 
BUSINESS_ID , 
NULL )
)); 
```

**Output**

```sql
CREATE UNIQUE INDEX gcc_plan_n2 ON gcc_plan.gcc_plan_t
 (( CASE WHEN ( enable_flag , 'Y' , business_id , NULL ) ));
```

**CASE statement**

CASE statement not supported in index creation.

**Input**

```sql
CREATE
UNIQUE index GCC_RSRC_ASSIGN_U1
ON GCC_PLAN.GCC_RSRC_ASSIGN_T ( 
(CASE
WHEN( ENABLE_FLAG = 'Y' AND ASSIGN_TYPE = '13' AND WORK_ORDER_ID IS NOT NULL )
THEN WORK_ORDER_ID
ELSE NULL
END)
);
```

**Output**

```sql
CREATE UNIQUE INDEX gcc_rsrc_assign_u1
ON gcc_plan.gcc_rsrc_assign_t
 (( ( CASE
WHEN ( enable_flag = 'Y' AND assign_type = '13' AND work_order_id IS NOT NULL )
THEN work_order_id
ELSE NULL END ))
); 
```
6.13.2.6 Views

A **view** is a logical table that is based on one or more tables or views. A view by itself contains no data.

In the source file, if the table names are not qualified with the schema name, then the target file is modified such that the table is also qualified with the same schema name as that of the view.

**Figure 6-13 Input - Views**

```
CREATE OR REPLACE VIEW schema1.v_view_name AS SELECT dict_code, code, dict_name, name
FROM tab1
WHERE BEAN_CODE = 'LOA_PERSONAL_ACCOUNT4PRTYCODE'
AND WORK_BT = ( SELECT MAX(WORK_DT) FROM tab2
WHERE BEAN_CODE = 'LOA_PERSONAL_ACCOUNT4PRTYCODE'
) AND WORK_BT = ( SELECT MAX(WORK_DT) FROM schema2.tab3
WHERE BEAN_CODE = 'LOA_PERSONAL_ACCOUNT4PRTYCODE'
)
```
6.13.2.7 Sequences

A sequence is an Oracle object used to generate a number sequence. This can be useful when you need to create an autonumber column to act as a primary key.

If MigSupportSequence is set to true (default), a sequence is created in the PUBLIC schema.

**NOTE**

- CACHE and ORDER cannot be migrated.
- In Oracle, the maximum value of MAXVALUE can be set to 999999999999999999999999999. In DWS, the maximum value of MAXVALUE can be set to 9223372036854775807.
- Before migrating a sequence, copy the content in the custom_scripts.sql file and paste it to execute the script in all the target databases. For details, see Custom DB Script Configuration and Migration Process.

### Sequence

**Input - CREATE SEQUENCE**

```sql
CREATE SEQUENCE GROUP_DEF_SEQUENCE
minvalue 1
maxvalue 100000000000000000000
start with 1152
increment by 1
```

**Output**

```sql
INSERT INTO PUBLIC.MIG_SEQ_TABLE (PUBLIC.MIG_SEQ_TABLE (```
Sequence with NOCACHE

Input - CREATE SEQUENCE with NOCACHE

CREATE SEQUENCE customers_seq
START WITH 1000
INCREMENT BY 1
NOCACHE
NOCYCLE;

Output

INSERT INTO PUBLIC.MIG_SEQ_TABLE (SCHEMA_NAME, SEQUENCE_NAME, START_WITH, INCREMENT_BY, MIN_VALUE, MAX_VALUE, CYCLE_I, CACHE, ORDER_I)
VALUES (UPPER( current_schema() ), UPPER( 'GROUP_DEF_SEQUENCE' ), 1152, 1, 1, 922372036854775807, FALSE, 20, FALSE);

CREATE SEQUENCE with Schema Name

Input - CREATE SEQUENCE with a specified schema name

CREATE SEQUENCE scott.seq_customers
START WITH 1000
INCREMENT BY 1
MINVALUE 1000  MAXVALUE 9999999999999999
CACHE 20  CYCLE ORDER;

**Output**

```sql
OUTPUT
INSERT INTO
    PUBLIC.MIG_SEQ_TABLE (  
        SCHEMA_NAME ,SEQUENCE_NAME ,START_WITH ,INCREMENT_BY ,MIN_VALUE ,MAX_VALUE ,CYCLE_I ,CACHE ,ORDER_I 
    )
VALUES (  
    UPPER( 'scott' ) ,UPPER( 'seq_customers' ) ,1000 ,1 ,1000 ,1000 ,999999999999999999999999999 ,TRUE ,20 ,FALSE 
    )
;
```

**CREATE SEQUENCE with Default Values**

**Input - SEQUENCE with a default value**

CREATE SEQUENCE seq_orders;

**Output**

```sql
OUTPUT
INSERT INTO
    PUBLIC.MIG_SEQ_TABLE (  
        SCHEMA_NAME ,SEQUENCE_NAME ,START_WITH ,INCREMENT_BY ,MIN_VALUE ,MAX_VALUE ,CYCLE_I ,CACHE ,ORDER_I 
    )
VALUES (  
    UPPER( current_schema() ) ,UPPER( 'seq_orders' ) ,1 ,1 ,1 ,999999999999999999999999999 ,TRUE ,20 ,FALSE 
    )
;
```

**NEXTVAL**

To migrate the NEXTVAL function, a custom function is provided for generating the next value based on `increment_by`, `max_value`, `min_value`, and `cycle`. During
the installation, this function should be created in all the databases where the migration is to be performed.

This function supports all versions of DWS.

NEXTVAL is a system function in Oracle and DWS does not implicitly support this function. To support this function, Migration Tool creates a NEXTVAL function in the PUBLIC schema. The PUBLIC.NEXTVAL function is used in the migrated statements.

NOTE

If MigSupportSequence is set to true, NEXTVAL is migrated to PUBLIC.NEXTVAL([schema].sequence').

If MigSupportSequence is set to false, NEXTVAL is migrated to NEXTVAL([schema].sequence').

Before migrating the NEXTVAL function, copy the content in the custom_scripts.sql file and paste it to execute the script in all the target databases. For details, see Custom DB Script Configuration and Migration Process.

Input - NEXTVAL

[schema.]sequence.NEXTVAL

Output

PUBLIC.nextval('[schema.]sequence')

Input - NEXTVAL

SELECT
    EMP_ID_SEQ.NEXTVAL INTO SEQ_NUM
FROM     dual
;

Output

SELECT
    PUBLIC.NEXTVAL ('EMP_ID_SEQ') INTO SEQ_NUM
FROM     dual
;

CURRVAL

To migrate the CURRVAL function, you can custom one to return the current value of a sequence. During the installation, this function should be created in all the databases where the migration is to be performed.

CURRVAL is a system function in Oracle and DWS does not implicitly support this function. To support this function, Migration Tool creates a CURRVAL function in the PUBLIC schema. The PUBLIC.CURRVAL function is used in the migrated statements.
If `MigSupportSequence` is set to `true`, CURRVAL is migrated to `PUBLIC.CURRVAL('[schema].sequence')`. If `MigSupportSequence` is set to `false`, CURRVAL is migrated to CURRVAL('[schema].sequence')

Before migrating the NEXTVAL function, copy the content in the `custom_scripts.sql` file and paste it to execute the script in all the target databases. For details, see Custom DB Script Configuration and Migration Process.

### Input - CURRVAL

\[
\text{[schema.]sequence.CURRVAL}
\]

### Output

\[
\text{currval('[schema.]sequence')}
\]

### Input - CURRVAL

```
INSERT INTO Line_items_tab (Orderno, Partno, Quantity) VALUES (Order_seq.CURRVAL, 20321, 3);
```

### Output

```
INSERT INTO Line_items_tab (Orderno, Partno, Quantity) SELECT PUBLIC.CURRVAL ('Order_seq'), 20321, 3;
```

### 6.13.2.8 Purge

In Oracle, `DROP TABLE` statement moves a table to the recycle bin. The `PURGE` statement is used to remove a table or index from recycle bin and release all of the space associated with the object. This statement is also used to remove the entire recycle bin.

The migrated query also does not contain `PURGE`.

**Figure 6-15** Input - Purge

```
Execute immediate 'Drop table table1 purge';
drop table test.emp purge;
```
6.13.2.9 Database Keywords

DWS keywords such as NAME, LIMIT, OWNER, KEY and CAST are supported by the Migration Tool by enclosing the keywords in double-quotes (".

Gauss Keyword (NAME, VERSION, LABEL, POSITION)

When keywords appear as aliases should be replaced with as and the keyword.

Input - (NAME, VERSION, LABEL, POSITION)

```sql
SELECT id, NAME, label, description
FROM (SELECT a.id id,
         b.NAME AS NAME,
         b.default_label label,
         ROW_NUMBER() OVER( ) ROW_ID
FROM CTP_ITEM A
LEFT OUTER JOIN CTP_ITEM_NLS B ON A.ID = B.ID
AND B.LOCALE = i_language
ORDER BY a.id ASC)
WHERE ROW_ID >= to_number(begNum)
AND ROW_ID < to_number(begNum) + to_number(fetchNum);

SELECT DISTINCT REPLACE(VERSION,' ','') ID, VERSION AS TEXT
FROM (SELECT T1.SOFTASSETS_NAME, T2.VERSION
      FROM SPMS_SOFT_ASSETS T1, SPMS_SYSSOFT_ASSETS T2
      WHERE T1.SOFTASSETS_ID = T2.SOFTASSETS_ID)
WHERE SOFTASSETS_NAME = I_SOFT_NAME;

SELECT COUNTRY, AMOUNT
FROM (SELECT " COUNTRY || " AMOUNT, '1' POSITION
      FROM DUAL)
ORDER BY POSITION;
```

Output

```sql
SELECT id, NAME, label, description FROM ( SELECT a.id b.NAME AS NAME,
         b.default_label AS label,
         ROW_NUMBER() OVER( ) OVER( ) OVER( ) ROW_ID
FROM CTP_ITEM A LEFT OUTER JOIN CTP_ITEM_NLS B ON A.ID = B.ID
AND B.LOCALE = i_language
ORDER BY a.id ASC)
WHERE ROW_ID >= to_number(begNum)
AND ROW_ID < to_number(begNum) + to_number(fetchNum);

SELECT DISTINCT REPLACE(VERSION, ' ', '') ID, VERSION AS TEXT
FROM (SELECT T1.SOFTASSETS_NAME, T2.VERSION
      FROM SPMS_SOFT_ASSETS T1, SPMS_SYSSOFT_ASSETS T2
      WHERE T1.SOFTASSETS_ID = T2.SOFTASSETS_ID)
WHERE SOFTASSETS_NAME = I_SOFT_NAME;

SELECT COUNTRY, AMOUNT
FROM (SELECT " COUNTRY || " AMOUNT, '1' POSITION
      FROM DUAL)
ORDER BY POSITION;
```
T1.SOFTASSETS_NAME
 ,T2.VERSION
 FROM
 SPMS_SOFT_ASSETS T1
 ,SPMS_SYSSOFT_ASSETS T2
 WHERE
 T1.SOFTASSETS_ID = T2.SOFTASSETS_ID
 )
 WHERE SOFTASSETS_NAME = I_SOFT_NAME ;

SELECT COUNTRY ,AMOUNT
 FROM ( SELECT '' COUNTRY || '' AMOUNT
       ,'1' AS POSITION
       FROM
       DUAL
     )
 ORDER BY
 POSITION
 ;

TEXT & YEAR

INPUT – (TEXT, YEAR)

SELECT
 NAME,
 VALUE,
 DESCRIPTION TEXT,
 JOINED YEAR,
 LIMIT
 FROM
 EMPLOYEE;

SELECT
 NAME,
 TEXT,
 YEAR,
 VALUE,
 DESCRIPTION,
 LIMIT
 FROM
 EMPLOYEE_DETAILS;

OUTPUT:

SELECT
 "NAME",
 VALUE,
 DESCRIPTION AS TEXT,
 JOINED AS YEAR,
 "LIMIT"
 FROM
 EMPLOYEE;

SELECT
 "NAME",
 "TEXT",
 "YEAR",
 VALUE,
 DESCRIPTION,
 "LIMIT"
 FROM
 EMPLOYEE_DETAILS;
NAME and LIMIT

**Input - DWS Keywords NAME and LIMIT**

```sql
CREATE TABLE NAME
  ( NAME VARCHAR2(50) NOT NULL
    , VALUE VARCHAR2(255)
    , DESCRIPTION VARCHAR2(4000)
    , LIMIT NUMBER(9)
  )
/*TABLESPACE users*/
pctfree 10 initrans 1 maxtrans 255
storage ( initial 256K next 256K
  minextents 1 maxextents
  unlimited );
SELECT NAME, VALUE, DESCRIPTION, LIMIT
FROM NAME;
```

**Output**

```sql
CREATE TABLE "NAME"
  ( "NAME" VARCHAR2 (50) NOT NULL
    , VALUE VARCHAR2 (255)
    , DESCRIPTION VARCHAR2 (4000)
    , "LIMIT" NUMBER (9)
  )
/*TABLESPACE users*/
pctfree 10 initrans 1 maxtrans 255
storage ( initial 256 K next 256 K
  minextents 1 maxextents
  unlimited );
SELECT "NAME", VALUE, DESCRIPTION, "LIMIT"
FROM "NAME";
```

**OWNER**

**Bulk Operations**

**Input - DWS Keyword OWNER using SELECT**

```sql
SELECT owner
FROM Test_Col;
```

**Output**

```sql
SELECT "OWNER"
FROM Test_Col;
```

**Input - DWS Keywords OWNER using DELETE**

```sql
DELETE FROM emp14
WHERE ename = 'Owner';
```

**Output**

```sql
DELETE FROM emp14
WHERE ename = 'Owner';
```
KEY

Blogic Operations

Input - DWS Keyword KEY

CREATE FUNCTION myfct RETURN VARCHAR2 parallel_enable IS res VARCHAR2 ( 200 ) ;
BEGIN
 res := 100 ;
 INSERT INTO emp18 RW ( RW.empno ,RW.ename ) SELECT
 res ,RW.ename KEY
 FROM
 emp16 RWN ;
 COMMIT ;
 RETURN res ;
END ;
/

Output

CREATE FUNCTION myfct RETURN VARCHAR2 IS res VARCHAR2 ( 200 ) ;
BEGIN
 res := 100 ;
 INSERT INTO emp18 ( empno ,ename ) SELECT
 res ,RW.ename "KEY"
 FROM
 emp16 RWN ;
 /* COMMIT; */
 null ;
 RETURN res ;
END ;

Range, Account and Language

In case Gauss keywords are used as Alias name for any column in SELECT list without defining AS keyword, need to use AS keyword to define Alias name.

Input

CREATE /*FORCE*/
VIEW SAD.FND_TERRITORIES_TL_V ( TERRITORY_CODE ,TERRITORY_SHORT_NAME ,LANGUAGE ,Account ,Range ,LAST_UPDATED_BY ,LAST_UPDATE_DATE ,LAST_UPDATE_LOGIN ,DESCRIPTION ,SOURCE_LANG ,ISO_NUMERIC_CODE ) AS SELECT
FROM
Primary Key, Unique Key

In case both constraints are declared on table creation, i.e. primary key and unique key, only primary key needs to consider for migration.
create table SD_WO.WO_DU_TRIGGER_REVENUE_T
(
    TRIGGER_REVENUE_ID NUMBER not null,
    PROJECT_NUMBER VARCHAR2(40),
    DU_ID NUMBER,
    STANDARD_MS_CODE VARCHAR2(100),
    TRIGGER_STATUS NUMBER,
    TRIGGER_MSG VARCHAR2(4000),
    BATCH_NUMBER NUMBER,
    PROCESS_STATUS NUMBER,
    ENABLE_FLAG CHAR(1) default 'Y',
    CREATED_BY NUMBER,
    CREATION_DATE DATE,
    LAST_UPDATE_BY NUMBER,
    LAST_UPDATE_DATE DATE
);

alter table SD_WO.WO_DU_TRIGGER_REVENUE_T
add constraint WO_DU_TRIGGER_REVENUE_PK primary key (TRIGGER_REVENUE_ID);
alter table SD_WO.WO_DU_TRIGGER_REVENUE_T
add constraint WO_DU_TRIGGER_REVENUE_N1 unique (DU_ID, STANDARD_MS_CODE);

6.13.2.10 Exceptions Feature

dup_val_on_index (Default configuration)

Input

CREATE OR REPLACE PACKAGE BODY test AS
PROCEDURE test1234(i1 int) IS out_flag VARCHAR(1);
BEGIN
    INSERT INTO tab1
    VALUES
    (i1, sysdate);
    RETURN;
EXCEPTION
    when no_data_found then
        out_flag := '1';
        DBMS_OUTPUT.PUT_LINE(out_flag);
        return;
    WHEN DUP_VAL_ON_INDEX then
        out_flag := '2';
        DBMS_OUTPUT.PUT_LINE(out_flag);
        return;
end test1234;
end test;
/
**Output**

```sql
CREATE OR REPLACE PROCEDURE test.test1234 ( i1 int ) IS out_flag VARCHAR( 1 ) ;
BEGIN
    INSERT INTO tab1
    VALUES ( i1 ,sysdate ) ;
    RETURN ;
EXCEPTION when no_data_found
then
    out_flag := '1' ;
    DBMS_OUTPUT.PUT_LINE ( out_flag ) ;
    return ;
WHEN UNIQUE_VIOLATION
THEN
    out_flag := '2' ;
    DBMS_OUTPUT.PUT_LINE ( out_flag ) ;
    return ;
END ;
/
```

**invalid_number (Default configuration)**

**Input**

```sql
CREATE OR REPLACE PACKAGE BODY test AS
PROCEDURE test123(i1 int) IS out_flag VARCHAR(1);
BEGIN
    INSERT INTO tab1
    VALUES
    (i1 / 0, sysdate);
RETURN;
EXCEPTION when zero_divide then out_flag := '1';
    DBMS_OUTPUT.PUT_LINE(out_flag);
    return;
WHEN invalid_number then out_flag := '2';
    DBMS_OUTPUT.PUT_LINE(out_flag);
    return;
WHEN OTHERS then out_flag := '-1';
    DBMS_OUTPUT.PUT_LINE(out_flag);
    return;
end test123;
end test;
/
```

**Output**

```sql
CREATE OR REPLACE PROCEDURE test.test123 ( i1 int ) IS out_flag VARCHAR( 1 ) ;
BEGIN
    INSERT INTO tab1
    VALUES ( i1 / 0 ,sysdate ) ;
    RETURN ;
EXCEPTION when zero_divide
then
    out_flag := '1' ;
    DBMS_OUTPUT.PUT_LINE ( out_flag ) ;
    return ;
WHEN invalid_number
then
    out_flag := '2' ;
    DBMS_OUTPUT.PUT_LINE ( out_flag ) ;
    return ;
WHEN OTHERS
THEN
    out_flag := '-1' ;
    DBMS_OUTPUT.PUT_LINE ( out_flag ) ;
    return ;
```
zero_divide (Default configuration)

Input
CREATE OR REPLACE PACKAGE BODY test AS
PROCEDURE test123(i1 int) IS out_flag VARCHAR(1);
BEGIN INSERT INTO tab1
VALUES
(i1 / 0, sysdate);
RETURN;
EXCEPTION when zero_divide then out_flag := '1';
DBMS_OUTPUT.PUT_LINE(out_flag);
return;
WHEN DUP_VAL_ON_INDEX THEN out_flag := '2';
DBMS_OUTPUT.PUT_LINE(out_flag);
return;
WHEN OTHERS THEN out_flag := '-1';
DBMS_OUTPUT.PUT_LINE(out_flag);
end test123;
end test;
/

Output
CREATE OR REPLACE PROCEDURE test.test123 ( i1 int ) IS out_flag VARCHAR( 1 ) ;
BEGIN
INSERT INTO tab1
VALUES ( i1 / 0 ,sysdate ) ;
RETURN ;
EXCEPTION when zero_divide
then
out_flag := '1' ;
DBMS_OUTPUT.PUT_LINE ( out_flag ) ;
return ;
WHEN UNIQUE_VIOLATION
THEN
out_flag := '2' ;
DBMS_OUTPUT.PUT_LINE ( out_flag ) ;
return ;
WHEN OTHERS
THEN
out_flag := '-1' ;
DBMS_OUTPUT.PUT_LINE ( out_flag ) ;
return ;
END ;
/

dup_val_on_index

Input
CREATE OR REPLACE PACKAGE BODY test AS
PROCEDURE test1234(i1 int) IS out_flag VARCHAR(1);
BEGIN INSERT INTO tab1
VALUES
(i1, sysdate);
RETURN;
EXCEPTION when no_data_found then out_flag := '1';
DBMS_OUTPUT.PUT_LINE(out_flag);
WHEN DUP_VAL_ON_INDEX THEN out_flag := '2';
DBMS_OUTPUT.PUT_LINE(out_flag);
return;
end test1234;

end test;
/

CREATE OR REPLACE PROCEDURE test.test1234 ( i1 int ) IS
out_flag VARCHAR( 1 ) ;
BEGIN
INSERT INTO tab1
VALUES ( i1 ,sysdate ) ;
RETURN ;
EXCEPTION when no_data_found
then
out_flag := '1' ;
DBMS_OUTPUT.PUT_LINE ( out_flag ) ;
return ;
WHEN OTHERS
THEN
IF sqlstate = 23505
THEN
out_flag := '2' ;
DBMS_OUTPUT.PUT_LINE ( out_flag ) ;
return ;
END IF ;
END ;
/

Input

CREATE OR REPLACE PACKAGE BODY test AS
PROCEDURE test123(i1 int) IS
out_flag VARCHAR(1);
BEGIN
INSERT INTO tab1
VALUES (i1 / 0, sysdate);
RETURN;
EXCEPTION when zero_divide then out_flag := '1';
DBMS_OUTPUT.PUT_LINE(out_flag);
return;
WHEN invalid_number then out_flag := '-1';
DBMS_OUTPUT.PUT_LINE(out_flag);
return;
WHEN OTHERS THEN out_flag := '-1';
DBMS_OUTPUT.PUT_LINE(out_flag);
return;
end test123;
end test;
/

Output

CREATE OR REPLACE PROCEDURE test.test123 ( i1 int ) IS
out_flag VARCHAR( 1 ) ;
BEGIN
INSERT INTO tab1
VALUES ( i1 / 0 ,sysdate ) ;
RETURN ;
EXCEPTION when DIVISION_BY_ZERO
then

out_flag := '1';
    DBMS_OUTPUT.PUT_LINE ( out_flag ) ;
return ;
WHEN INVALID_PARAMETER_VALUE
then
    out_flag := '2';
    DBMS_OUTPUT.PUT_LINE ( out_flag ) ;
return ;
WHEN OTHERS
THEN
    out_flag := '-1';
    DBMS_OUTPUT.PUT_LINE ( out_flag ) ;
return ;
END ;
/

zero_divide

Input
CREATE OR REPLACE PACKAGE BODY test AS
PROCEDURE test123(i1 int) IS out_flag VARCHAR(1);
BEGIN INSERT INTO tab1
VALUES
(i1 / 0, sysdate);
RETURN;
EXCEPTION when zero_divide then out_flag := '1';
DBMS_OUTPUT.PUT_LINE(out_flag);
return;
WHEN DUP_VAL_ON_INDEX THEN out_flag := '2';
DBMS_OUTPUT.PUT_LINE(out_flag);
return;
WHEN OTHERS THEN out_flag := '-1';
DBMS_OUTPUT.PUT_LINE(out_flag);
return;
end test123;
end test;
/

Output
CREATE OR REPLACE PROCEDURE test.test123 ( i1 int ) IS out_flag VARCHAR( 1 ) ;
BEGIN
    INSERT INTO tab1
    VALUES ( i1 / 0 ,sysdate ) ;
    RETURN ;
EXCEPTION when DIVISION_BY_ZERO
then
    out_flag := '1' ;
    DBMS_OUTPUT.PUT_LINE ( out_flag ) ;
    return ;
WHEN OTHERS
THEN
    IF sqlstate = 23505
        THEN
            out_flag := '2';
            DBMS_OUTPUT.PUT_LINE ( out_flag ) ;
            return ;
        ELSE
            out_flag := '-1';
            DBMS_OUTPUT.PUT_LINE ( out_flag ) ;
            return ;
        END IF ;
    END ;
/

6.13.3 DML

6.13.3.1 Overview
This section contains the migration syntax for migrating Oracle DML. The migration syntax decides how the supported keywords/features are migrated.

6.13.3.2 SELECT

Overview
The Oracle SELECT statement starts a query, with an optional ORDER BY clause, used to retrieve records from one or more tables in a database.

Input - SELECT

```
SELECT col1, col2
FROM tab1;
```

Output

```
SELECT col1, col2
FROM tab1;
```

Order of Clauses

The HAVING clause must be present after the GROUP BY clause whereas, Oracle allows HAVING before or after the GROUP BY clause. In the target, the HAVING clause is moved to after the GROUP BY clause.

Figure 6-17 Input - Order of Clauses

```
SELECT
  DEPTNO
 , COUNT ( "" )
 , SUM ( SAL )
FROM
EMP
WHERE
  JOB = "CLERK"
GROUP BY
  DEPTNO
HAVING
  SUM ( SAL ) >= 500
ORDER BY
  DEPTNO
```
Extended Group By Clause

The GROUP BY clause can be specified if we want the database to group the selected rows based on the value of expr(s). If this clause contains CUBE, ROLLUP or GROUPING SETS extensions, then the database produces super-aggregate groupings in addition to the regular groupings. These features are not available in DWS and these have been addressed using the UNION ALL operator.
GROUPING_ID with ROLLUP

GROUPING_ID returns a number that corresponds to the GROUPING bit vector associated with a row. GROUPING_ID is applicable only in a SELECT statement containing a GROUP BY extension, such as ROLLUP, and a GROUPING function. In queries with multiple GROUP BY expressions, determining the GROUP BY level of a particular row requires many GROUPING functions, which can lead to complicated SQLs. In such scenarios, GROUPING_ID is useful.
**Table Name Inside Brackets**

Table names need not be specified within parentheses whereas, Oracle accepts the parentheses.

**Figure 6-21** Input - Table Name Inside Brackets

```sql
SELECT 
FROM
( emp ) e 
WHERE 
e.deptno = 1 
;
```

**Figure 6-22** Output - Table Name Inside Brackets

```sql
SELECT DISTINCT a.item_id id, 
a.menu_id parent_id,a.serialno menu_order 
FROM ctp_menu_item_rel a WHERE 
a.item_id IN(SELECT UNIQUE id FROM ctp_temp_item_table);
```

**UNIQUE Keyword**

Unique is migrated as Distinct keyword.

**Input - SELECT UNIQUE**

```sql
SELECT UNIQUE a.item_id, 
a.menu_id parent_id,a.serialno menu_order 
FROM ctp_menu_item_rel a WHERE 
a.item_id IN(SELECT UNIQUE id FROM ctp_temp_item_table);
```

**Output**

```sql
SELECT DISTINCT a.item_id id, 
a.menu_id parent_id,a.serialno menu_order 
FROM ctp_menu_item_rel a WHERE 
a.item_id IN(SELECT UNIQUE id FROM ctp_temp_item_table);
```

**USERENV**

**Input - CLIENT_INFO**

Returns user session information.

```sql
SELECT 1 
FROM   sp_ht ht 
WHERE  ht.hth = pi_contract_number /* AND ht.contract_status = 2 --delete by leinian 2014-03-03(ECO) */ 
AND ht.contract_status IN ( 1, 2 ) /* add by leinian 2014-03-20(ECO) */ 
AND Nvl(ht.s3_pilot_flag, 'N') = 'N' 
AND NOT EXISTS (SELECT 1 
FROM   asms.asms_lookup_values alv 
WHERE  alv.type_code = 'HTLX_LOAN' 
AND ht.htlx = alv.code) 
AND ht.duty_erp_ou_id = To_number(Nvl(Rtrim(Ltrim(Substr(Userenv( 
```

```sql
Data Warehouse Service Tool Guide

Issue 03 (2019-08-08) Copyright © Huawei Technologies Co., Ltd. 491
After the function in the package is converted, the function tag after end is not deleted. The svproduct_is_for_pa function in 4. sad_lookup_contract_pkg.bdy is used.

**USERENV('CLIENT_INFO')**

After the function in the package is converted, the function tag after end is not deleted. The svproduct_is_for_pa function in 4. sad_lookup_contract_pkg.bdy is used.

**USERENV('CLIENT_INFO')**

USERENV used inside procedure. Migration process getting failed by Tool.

```
SELECT 1
FROM   sp_ht ht
WHERE  ht.hth = pi_contract_number
       AND ht.contract_status = 2 /* delete by leinian 2014-03-03(ECO) */
       AND ht.contract_status IN ( 1,
       2 ) /* add by leinian 2014-03-20(ECO) */
       AND Nvl( ht.s3_pilot_flag, 'N' ) = 'N'
       AND NOT EXISTS ( SELECT
                    1
               FROM   asms.asms_lookup_values alv
               WHERE  alv.type_code = 'HTLX_LOAN'
                    AND ht.htlx = alv.code )
       AND ht.duty_erp_ou_id = To_number( Nvl( Rtrim( Ltrim( Substr( Userenv( 'client_info' ), 1, 8 ) ) ) ,218 ) )
       AND ht.source_code = 'ECONTRACT'
       AND ht.needing_engineering_service IS NOT NULL
       AND ht.khm != '28060'
       AND ht.htlx != '111' ;
```

**Input**

Error message: client_info argument for USERENV function is not supported by the Migration Tool.
CREATE OR REPLACE PACKAGE BODY SAD.sad_lookup_contract_pkg IS
FUNCTION svproduct_is_for_pa(pi_contract_number IN VARCHAR2) RETURN VARCHAR2 IS
    l_flag VARCHAR2(1) := 'N';
BEGIN
    FOR rec_lookup IN (SELECT 1
        FROM asms.asms_lookup_values alv
        WHERE alv.type_code = 'HTLX_LOAN'
        AND alv.duty_erp_ou_id = to_number(nvl(rtrim(ltrim(substr(userenv('client_info'), 1, 8))), 218))
    ) LOOP
        l_flag := 'Y';
    END LOOP;
RETURN l_flag;
END svproduct_is_for_pa;
END sad_lookup_contract_pkg;
/

Output
CREATE OR replace FUNCTION sad_lookup_contract_pkg.Svproduct_is_for_pa (pi_contract_number IN VARCHAR2)
RETURN VARCHAR2 IS
    l_flag VARCHAR2 ( 1 ) := 'N';
BEGIN
    FOR rec_lookup IN (SELECT 1
        FROM asms.asms_lookup_values alv
        WHERE alv.type_code = 'HTLX_LOAN'
        AND alv.duty_erp_ou_id = to_number(nvl(rtrim(ltrim(substr(userenv('client_info'), 1, 8))), 218))
    ) LOOP
        l_flag := 'Y';
    END LOOP;
RETURN l_flag;
END;
/

6.13.3.3 INSERT

Overview

The Oracle INSERT statement is used to insert a single record or multiple records into a table.

INSERT ALL

The Oracle INSERT ALL statement is used to add multiple rows using a single INSERT statement. The rows can be inserted into either a single table or multiple tables. The target query is converted as a common table expression (CTE).
**Figure 6-23 Input - INSERT ALL**

```
INSERT ALL INTO ap_cust
VALUES (customer_id, program_id, delivered_date)
INTO ap_orders (ord_dt, Prg_id)
VALUES (order_date, program_id)

SELECT program_id, delivered_date, customer_id, order_date
FROM ORDER
WHERE deptno = 10
```
**INSERT FIRST**

The Oracle **INSERT FIRST** is used to execute an insert statement associated with the first condition that is true; other statements are ignored. The target query is converted as a common table expression (CTE).
**Figure 6-25** Input - Insert First

```sql
INSERT
  FIRST WHEN deptno <= 10
  THEN INTO emp12 WHEN comm > 500
  THEN INTO emp13
  SELECT empno,
       ,ename
       ,job
       ,mgr
       ,hiredate
       ,sal
       ,comm
       ,deptno
FROM emp
WHERE deptno IS NOT NULL
```
Figure 6-26 Output - Insert First

```
WITH Sel AS (  
  SELECT  
    ROW_NUMBER() OVER() AS Ins_First_RN  
    ,empno  
    ,ename  
    ,job  
    ,mgr  
    ,hiredate  
    ,sal  
    ,comm  
    ,deptno  
  FROM  
  emp  
  WHERE  
  deptno IS NOT NULL
)  
,ins1 AS (  
  INSERT INTO  
  emp12 (  
    SELECT  
      empno  
      ,ename  
      ,job  
      ,mgr  
      ,hiredate  
      ,sal  
      ,comm  
      ,deptno  
    FROM  
    Sel  
    WHERE  
    deptno <= 10  
  )  
  returning 1
)  
INSERT INTO  
emp13 (  
  SELECT  
    empno  
    ,ename  
    ,job  
    ,mgr  
    ,hiredate  
    ,sal  
    ,comm  
    ,deptno  
  FROM  
  (  
    SELECT  
      *  
    FROM  
    Sel  
    WHERE  
    comm > 500  
  )  
  s1  
  LEFT JOIN (  
    SELECT  
      Ins_first_RN  
    FROM  
    Sel  
    WHERE  
    deptno <= 10  
  )  
  s2  
  ON s1.Ins_first_RN = s2.Ins_first_RN  
WHERE  
  s2.Ins_first_RN IS NULL
```
INSERT with Table Alias

The Oracle table aliases are used to clarify and improve readability when referring to a table in a query by assigning it a name or code. INSERT with table alias can be used with INSERT INTO statement. The tool supports the migration of INSERT INTO statements with table alias.

● Blogic Operations

Input - INSERT with Table Alias

```sql
CREATE OR REPLACE FUNCTION myfct RETURN VARCHAR2 IS res VARCHAR2 (200); BEGIN res := 100; INSERT INTO emp18 RW (RW.empno,RW.ename) SELECT res,RWN.ename FROM emp16 RWN; COMMIT; RETURN res; END;/
```

Output

```sql
CREATE OR REPLACE FUNCTION myfct RETURN VARCHAR2 IS res VARCHAR2 (200); BEGIN res := 100; INSERT INTO emp18 (empno,ename) SELECT res,RWN.ename FROM emp16 RWN; /* COMMIT ';' */ null; RETURN res; END;/
```

● Bulk Operations

Input - INSERT with Table Alias

```sql
INSERT INTO Public.emp14 ats (ats.empno,ats.ename) VALUES (3,'Categories');
```

Output

```sql
INSERT INTO Public.emp14 (empno ,ename ) SELECT 3 , 'Categories' ;
```

Input - INSERT with Table Alias

```sql
INSERT INTO "abc"."emp18" wmc (
wmc.empno
, wmc.ename
) SELECT
    wmc.empno
,wm_concat (wmc.ename) AS eName
FROM
emp16 wmc
GROUP BY
empno
;

Output

INSERT
INTO
  "abc" . "emp18" ( empno , ename
) SELECT
    wmc.empno
 ,STRING_AGG ( wmc.ename
 ,'
') AS eName
FROM
emp16 wmc
GROUP BY
empno
;

Input - INSERT with Table Alias

INSERT
INTO
  emp14 "TABLE" ( "TABLE" .empno , ename
) SELECT
    empno
 ,ename
FROM
emp12
WHERE
    emp12.salary > ( SELECT
        MAX( salary )
FROM
emp13 "TABLE"
WHERE
    "TABLE" .empno > 5
 )
;

Output

INSERT
INTO
  emp14 ( empno , ename
) SELECT
    empno
 ,ename
FROM
emp12
WHERE
    emp12.salary > ( SELECT
        MAX( salary )
FROM
emp13 "TABLE"
WHERE
    "TABLE" .empno > 5
 )
6.13.3.4 MERGE

**MERGE** is an ANSI-standard SQL syntax operator used to select rows from one or more sources for updating or inserting into a table or view. The conditions to update or insert to the target table or view can be specified.

Currently, this 6.5.0 and later feature is supported by DWS. The Migration Tool uses multiple approaches to migrate **MERGE** into DWS compatible SQL.

The configuration parameter **mergeImplementation**:

- Is, by default, set to **WITH**. In this option, the target query is converted as a common table expression (CTE).

*Figure 6-27 Input - MERGE*
Can also be set to **SPLIT**. In this option, the **MERGE** statement is split into multiple **INSERT** and **UPDATE** statements.

Figure 6-29 Input - MERGE

```sql
MERGE INTO employees01 e
USING (SELECT empid, ename, startdate, address
      FROM hr_records
      WHERE empid > 100) h
ON (e.id = h.empid)
WHEN MATCHED THEN
  UPDATE SET e.address = h.address,
           e.ename = h.ename
WHEN NOT MATCHED THEN
  INSERT (empid, ename, startdate, address)
VALUES (h.empid, h.ename, h.startdate, h.address);
```
6.13.4 Pseudo Columns

6.13.4.1 Overview

This section contains the migration syntax for migrating Oracle Pseudo Columns. The migration syntax decides how the supported keywords/features are migrated.

A pseudocolumn is similar to a table column, but is not actually stored in the table. User can select from pseudocolumns, but cannot insert, update, or delete their values.

6.13.4.2 ROWID

The ROWID pseudocolumn returns the address of a specific row.

---

**Figure 6-30 Output - MERGE**

```sql
UPDATE employees01 e
SET e.address = h.address
  , e.ename = h.ename
FROM ( SELECT empid, ename, startdate, address
         FROM hr_records
         WHERE empid > 100
       ) h
WHERE e.id = h.empid;

INSERT INTO employees01 ( empid, ename, startdate, address )
SELECT h.empid, h.ename, h.startdate, h.address
FROM ( SELECT empid, ename, startdate, address
         FROM hr_records
         WHERE empid > 100
       ) h LEFT OUTER JOIN employees01 e
ON e.id = h.empid
WHERE e.id IS NULL;
```
6.13.4.3 ROWNUM

**ROWNUM**: For each row returned by a query, the ROWNUM pseudocolumn returns a number indicating the order in which Oracle selects the row from a table or set of joined rows. The first row selected has a ROWNUM of 1, the second has 2, and so on.

**Input-ROWNUM with UPDATE**

On UPDATE, if ROWNUM with some value (integer) is used, system will UPDATE those many records according to the operator used near ROWNUM accordingly.

```
UPDATE SCMS_MSGPOOL_LST
SET MSG_STD = '11'
WHERE UNISEQNO = IN_OUNISEQNO
AND MSG_TYP1 IN ('MT103', 'MT199')
AND ROWNUM = 1;
```

**Output**

```
UPDATE SCMS_MSGPOOL_LST
SET MSG_STD = '11'
WHERE (xc_node_id,ctid) in (select xc_node_id, ctid
from SCMS_MSGPOOL_LST
where UNISEQNO = IN_OUNISEQNO
```
Input-ROWNUM with DELETE

On DELETE, if ROWNUM with some value (integer) is used, system will DELETE those many records according to the operator used near ROWNUM accordingly.

```
delete from test1
where c1='abc' and rownum = 1;
```

Output

```
delete from test1 where (xc_node_id,ctid) in (select xc_node_id, ctid from test1 where c1='abc' limit 1);
```

Input - UPDATE with ROWNUM

Migrated scripts of UPDATE and DELETE with ROWNUM contain LIMIT, which is not supported by Gauss.

```
UPDATE SCMS_MSGPOOL_LST
SET MSG_STD = '11'
WHERE UNISEQNO = IN_OUNISEQNO
AND MSG_TYP1 IN ('MT103', 'MT199')
AND ROWNUM = 1;
```

Output

```
UPDATE SCMS_MSGPOOL_LST
SET MSG_STD = '11'
WHERE (xc_node_id, ctid) = ( SELECT xc_node_id, ctid
    FROM SCMS_MSGPOOL_LST
    WHERE UNISEQNO = IN_OUNISEQNO
    AND MSG_TYP1 IN ('MT103', 'MT199')
    LIMIT 1
);
```

Input - DELETE with ROWNUM

```
DELETE FROM SPMS_APP_PUBLISH
WHERE NOVA_NO = IN_NOVA_NO
AND DELIVERY_TYPE = '1'
AND PUBLISH_DATE = IN_PUBLISH_DATE
AND ROWNUM = 1;
```

Output

```
DELETE FROM SPMS_APP_PUBLISH
WHERE (xc_node_id, ctid) IN (SELECT xc_node_id, ctid
    FROM SPMS_APP_PUBLISH
    WHERE NOVA_NO = IN_NOVA_NO
    AND DELIVERY_TYPE = '1'
    AND PUBLISH_DATE = IN_PUBLISH_DATE
    LIMIT 1
);
```

6.13.5 Outer Join

This section contains the migration syntax for migrating Oracle Outer Joins. The migration syntax decides how the supported keywords/features are migrated.

An **outer join** returns all rows that satisfy the join condition. In addition, it returns rows from one table for which no rows from the other satisfy the join condition. In Oracle,
● Left outer join of tables A and B returns all rows from A and rows that satisfy the join condition by applying the outer join operator (+) to all columns of B in the \texttt{WHERE} conditions.

● Right outer join of tables A and B returns all rows from B and rows that satisfy the join condition by applying the outer join operator (+) to all columns of A in the \texttt{WHERE} condition.

DWS do not support "+" operator. This operator is addressed using \texttt{LEFT OUTER JOIN} and \texttt{RIGHT OUTER JOIN} keywords.

\textbf{Figure 6-35} Input - Outer Join

\begin{verbatim}
SELECT empno ,ename ,job ,dname ,loc 
FROM emp ,dept 
WHERE emp.deptno = dept.deptno (+) 
AND salary > 50000;
\end{verbatim}

\textbf{Figure 6-36} Output - Outer Join

\begin{verbatim}
SELECT empno ,ename ,job ,dname ,loc 
FROM emp LEFT OUTER JOIN dept 
ON emp.deptno = dept.deptno 
WHERE salary > 50000;
\end{verbatim}

\textbf{6.13.6 OUTER QUERY (+)}

Join is supported in Gauss 18.2.0. So config parameter is added.

It depends on the following configuration item:

\texttt{supportJoinOperator=false}

\texttt{Input-OUTER QUERY(+)}

\begin{verbatim}
SELECT PP.PUBLISH_NO 
FROM SPMS_PARAM_PUBLISH PP 
WHERE PP.PUBLISH_ID(+) = TB2.PUBLISH_ID;
\end{verbatim}

\begin{verbatim}
SELECT I.APP_CHNAME, I.APP_SHORTNAME 
FROM SPMS_PARAM_PUBLISH PP 
WHERE PP.PUBLISH_ID(+) = TB2.PUBLISH_ID;
\end{verbatim}

\begin{verbatim}
SELECT I.APP_CHNAME, I.APP_SHORTNAME 
FROM SPMS_PARAM_PUBLISH PP 
WHERE PP.PUBLISH_ID(+) = TB2.PUBLISH_ID;
\end{verbatim}
FROM SPMS_APPVERSION SA, SPMS_APP_INFO I
WHERE SA.APP_ID = I.APP_ID(+)
AND SA.DELIVERY_USER = IN_USERID
ORDER BY APPVER_ID DESC ;

Output
SELECT
PP.PUBLISH_NO
FROM
SPMS_PARAM_PUBLISH PP
WHERE
PP.PUBLISH_ID (+) = TB2.PUBLISH_ID
;

SELECT
I.APP_CHNAME
, I.APP_SHORTNAME
FROM
SPMS_APPVERSION SA
, SPMS_APP_INFO I
WHERE
SA.APP_ID = I.APP_ID (+)
AND SA.DELIVERY_USER = IN_USERID
ORDER BY
APPVER_ID DESC
;

6.13.7 CONNECT BY

Input-CONNECT BY

select id from city_branch start with id=roleBranchId connect by prior id=parent_id;

SELECT T.BRANCH_LEVEL, t.ID
FROM city_branch c
WHERE (c.branch_level = '1' OR T.BRANCH_LEVEL = '2')
AND (T.SIGN = '1' OR T.SIGN = '4' OR T.SIGN = '8')
AND T.STATUS = '1'
START WITH c.ID = I_BRANCH_ID
CONNECT BY c.ID = PRIOR c.parent_id
ORDER BY c.branch_level DESC ;

Output
WITH RECURSIVE migora_cte AS (
SELECT
id
,1 AS LEVEL
FROM
city_branch
WHERE
id = roleBranchId
UNION
ALL SELECT
mig_ora_cnte_join_alias.id
,mig_ora_cnte_tab_alias.LEVEL + 1 AS LEVEL
FROM
migora_cnte mig_ora_cnte_tab_alias INNER JOIN city_branch mig_oracle_cnte_join_alias
ON mig_ora_cnte_tab_alias.id = mig_ora_cnte_join_alias.parent_id
)
SELECT
id
FROM
migora_cnte
ORDER BY
LEVEL
;
WITH RECURSIVE migora_cte AS (  
SELECT  
  BRANCH_LEVEL  
, ID  
, SIGN  
, STATUS  
, parent_id  
, 1 AS LEVEL  
FROM  
city_branch c  
WHERE  
c.ID = l_BRANCH_ID  
UNION  
ALL SELECT  
  c.BRANCH_LEVEL  
, c.ID  
, c.SIGN  
, c.STATUS  
, c.parent_id  
, mig_ora_cte_tab_alias.LEVEL + 1 AS LEVEL  
FROM  
migora_cte mig_ora_cte_tab_alias INNER JOIN city_branch c  
ON c.ID = mig_ora_cte_tab_alias.parent_id  
) SELECT  
  BRANCH_LEVEL  
, ID  
FROM  
migora_cte c  
WHERE  
  (  
c.branch_level = '1'  
OR T.BRANCH_LEVEL = '2'  
)  
AND (T.SIGN = '1' OR T.SIGN = '4' OR T.SIGN = '8')  
AND T.STATUS = '1'  
ORDER BY  
c.branch_level DESC  
)

Input - CONNECT BY multiple tables

It says how each child relates to its parent. This is defined using the CONNECT BY .. PRIOR clause, which defines how the current row (child) relates to a prior row (parent).

SELECT DISTINCT a.id menuId,  
F.name menuName,  
a.status menuState,  
a.parent_id menuParentId,  
'1' menuPrivilege,  
a.serialNo menuSerialNo  
FROM CTP_MENU a, CTP_MENU_NLS F  
START WITH a.serialno in (1, 2, 3)  
CONNECT BY a.id = PRIOR a.parent_id  
AND f.locale = Language  
AND a.id = f.id  
ORDER BY menuId, menuParentId;

Output

WITH RECURSIVE migora_cte AS (  
SELECT pr.service_product_id  
, t.enabled_flag  
, pr.operation_id  
, pr.enabled_flag  
, pr.product_code  
, 1 AS LEVEL  
FROM asms.cppsv_operation_sort t  
, asms.cppsv_product_class pr  

WHERE level_id = 3
AND properation_id = t.operation_id(+)
UNION ALL
SELECT pr.service_product_id
 , t.enabled_flag
 , pr.operation_id
 , pr.enabled_flag
 , pr.product_code
 , mig_ora_cte_tab_alias.LEVEL + 1 AS LEVEL
FROM migora_c te mig_ora_cte_tab_alias
 , asms.cppsv_operation_sort t
 , asms.cppsv_product_class pr
WHERE mig_ora_cte_tab_alias.service_product_id = pr.service_product_father_id
AND properation_id = t.operation_id(+)
)
SELECT pr.service_product_id
FROM migora_c te
WHERE nvl( UPPER( enabled_flag ) , 'Y' ) = 'Y'
AND nvl( enabled_flag , 'Y' ) = 'Y'
AND pr.product_code = rec_product1.service_product_code
ORDER BY LEVEL;

6.13.8 System Functions

6.13.8.1 Overview

This section contains the migration syntax for migrating Oracle system functions. The migration syntax decides how the supported keywords/features are migrated.

6.13.8.2 Date Functions

This section describes the following date functions:

- **ADD_MONTHS**
- **DATE_TRUNC**
- **LAST_DAY**
- **MONTHS_BETWEEN**
- **SYSTIMESTAMP**

**ADD_MONTHS**

ADD_MONTHS is an Oracle system function and is not implicitly supported by DWS.

**NOTE**

Before using this function, perform the following operations:

1. Create and use the MI G ORA_ EXT schema.
2. Copy the contents of the custom script and execute the script in all target databases for which migration is to be performed. For details, see Custom DB Script Configuration and Migration Process.

ADD_MONTHS returns a date with month.

- Data type of the **date** parameter is DATETIME.
- Data type of the **integer** parameter is INTEGER.

The return type is DATE.

**Input - ADD_MONTHS**
SELECT
    TO_CHAR( ADD_MONTHS ( hire_date ,1 ) , 'DD-MON-YYYY' ) "Next month"
FROM
    employees
WHERE
    last_name = 'Baer';

Output
SELECT
    TO_CHAR( MIG_ORA_EXT.ADD_MONTHS ( hire_date ,1 ) , 'DD-MON-YYYY' ) "Next month"
FROM
    employees
WHERE
    last_name = 'Baer';

DATE_TRUNC

The DATE_TRUNC function returns a date with the time portion of the day truncated to the unit specified by the format model **fmt**.

Input
select trunc(to_char(trunc(add_months(sysdate,-12), 'MM'), 'YYYYMMDD')/100) into v_start_date_s from dual;
select trunc(to_char(trunc(sysdate,'mm'), 'YYYYMMDD')/100) into v_end_date_e from dual;
ID_MNTH>=TRUNC(TO_CHAR(ADD_MONTHS(to_date(to_char(''||
v_curr_date||'), ''YYYYMMDD''), -12), ''YYYYMMDD'')/100))
AND ID_MNTH>=TRUNC(TO_CHAR(ADD_MONTHS(to_date(to_char(''||
v_curr_date||'), ''YYYYMMDD''), -12), ''YYYYMMDD'')/100))
select TRUNC(to_char(add_months(trunc(TO_DATE(TO_CHAR(P_DATE),'YYYYMMDD'),'MM')-1,-2),'YYYYMMDD')/100) INTO START_MONTH from dual;
select TRUNC(TO_CHAR(trunc(TO_DATE(TO_CHAR(P_DATE),'YYYYMMDD'),'MM')-1,'YYYYMMDD')/100) INTO END_MONTH from dual;

Output
SELECT Trunc(To_char(Date_trunc ('MONTH', mig_ora_ext.Add_months (SYSDATE, -12)) , 'YYYYMMDD') / 100)
INTO   v_start_date_s
FROM   dual;
SELECT Trunc(To_char(Date_trunc ('MONTH', SYSDATE), 'YYYYMMDD') / 100)
INTO   v_end_date_e
FROM   dual;
SELECT Trunc(To_char(mig_ora_ext.Add_months (Date_trunc ('MONTH', To_date(To_char(p_date), 'YYYYMMDD')) - 1 , -2), 'YYYYMMDD') / 100)
INTO   start_month
FROM   dual;
SELECT Trunc(To_char(Date_trunc ('MONTH', To_date(To_char(p_date), 'YYYYMMDD')) - 1, 'YYYYMMDD') / 100)
INTO   end_month
FROM   dual;

LAST_DAY

The Oracle LAST_DAY function returns the last day of the month based on a date value.

LAST_DAY(date)
The return type is always DATE, regardless of the data type of the date.

LAST_DAY is an Oracle system function and is not implicitly supported by DWS. To support this function, Migration Tool creates a LAST_DAY function in the MIG_ORA_EXT schema. The migrated statements will use the new function MIG_ORA_EXT.LAST_DAY as shown in the following example.

### Note

Before using this function, perform the following operations:

1. Create and use the MIG_ORA_EXT schema.

2. Copy the contents of the custom script and execute the script in all target databases for which migration is to be performed. For details, see Custom DB Script Configuration and Migration Process.

### Input - LAST_DAY

```sql
SELECT to_date('01/07/' || to_char(sysdate, 'YYYY') , 'dd/mm/yyyy') FIRST
, last_day( to_date('01/07/' || to_char(sysdate, 'YYYY') , 'dd/mm/yyyy') ) last__day
FROM dual;
```

### Output

```sql
SELECT to_date ('01/07/' || to_char(sysdate, 'YYYY') , 'dd/mm/yyyy') FIRST
, MIG_ORA_EXT.LAST_DAY (to_date ('01/07/' || to_char(sysdate, 'YYYY') , 'dd/mm/yyyy')) last__day
FROM dual;
```

### MONTHS_BETWEEN

The MONTHS_BETWEEN function returns the number of months between two dates.

MONTHS_BETWEEN is an Oracle system function and is not implicitly supported by DWS. To support this function, Migration tool creates a MONTHS_BETWEEN function in the MIG_ORA_EXT schema. The migrated statements will use the new function MIG_ORA_EXT.MONTHS_BETWEEN as shown in the following example.

### Note

Before using this function, perform the following operations:

1. Create and use the MIG_ORA_EXT schema.

2. Copy the contents of the custom script and execute the script in all target databases for which migration is to be performed. For details, see Custom DB Script Configuration and Migration Process.

### Input - MONTHS_BETWEEN

```sql
Select Months_Between(to_date('2017-06-20', 'YYYY-MM-DD'), to_date('2011-06-20', 'YYYY-MM-DD')) from dual;
```

### Output

```sql
Select MIG_ORA_EXT.MONTHS_BETWEEN(to_date('2017-06-20', 'YYYY-MM-DD'), to_date('2011-06-20', 'YYYY-MM-DD')) from dual;
```
The SYSTIMESTAMP function returns the system date, including fractional seconds and time zone, of the system on which the database resides. The return type is TIMESTAMP WITH TIME ZONE.

Figure 6-37 Input - SYSTIMESTAMP

```sql
SELECT SYSTIMESTAMP
FROM tab1;
```

Figure 6-38 Output - SYSTIMESTAMP

```sql
SELECT CURRENT_TIMESTAMP
FROM tab1;
```

6.13.8.3 LOB Functions

This section contains information on the LOB Functions. It is organized as follows:

- `DBMS_LOB.APPEND`
- `DBMS_LOB.COMPRESS`
- `DBMS_LOB.INSTR`
- `DBMS_LOB.SUBSTR`

DBMS_LOB.APPEND

DBMS_LOB.APPEND function appends the contents of a source LOB to a specified LOB.

**Input - DBMS_LOB.APPEND**

```sql
[sys.]dbms_lob.append(o_menuxml, to_clob('DSJKSDAJKSFDA'));
```

**Output**

```sql
o_menuxml := CONCAT(o_menuxml, CAST('DSJKSDAJKSFDA' AS CLOB));
```

**Input - DBMS_LOB.APPEND**

```sql
CREATE OR REPLACE PROCEDURE append_example IS clobSrc CLOB;
clobDest CLOB;
BEGIN
SELECT clobData INTO clobSrc
FROM myTable
WHERE id = 2;
SELECT
```
### clobData INTO clobDest

FROM

myTable

WHERE

id = 2 ;
readClob ( 1 ) ;
DBMS_LOB.APPEND ( clobDest ,clobSrc ) ;
readClob ( 1 ) ;
END append_example ;
/

---

### Output

CREATE OR REPLACE PROCEDURE append_example IS clobSrc CLOB ; clobDest CLOB ;
BEGIN
SELECT
clobData INTO clobSrc
FROM
myTable
WHERE
id = 2 ;
SELECT
clobData INTO clobDest
FROM
myTable
WHERE
id = 1 ;
readClob ( 1 ) ;
clobDest := CONCAT( clobDest ,clobSrc ) ;
readClob ( 1 ) ;
end ;
/

---

### DBMS_LOB.COMpare

DBMS_LOB.COMpare function compares the full/partial content of two LOBs. **DBMS_LOB.COMpare** is an Oracle system function and is not implicitly supported in DWS. To support this feature, the Migration tool creates a MIG_CLOB_COMPARE function in the MIG_ORA_EXT schema. The migrated statements will use this new function MIG_ORA_EXT.MIG_CLOB_COMPARE as shown in the SQL examples.

**COMPARE in SQL**

**Input - DBMS_LOB.COMpare in SQL**

```sql
SELECT a.empno ,dbms_lob.compare ( col1 ,col2 ) FROM emp a ,emp b ;
```

**Output**

```sql
SELECT a.empno ,MIG_ORA_EXT.MIG_CLOB_COMPARE ( col1 ,col2 ) FROM emp a ,emp b ;
```

**Input - DBMS_LOB.COMpare in SQL with CREATE TABLE using 5 parameters**

```sql
CREATE TABLE abc nologging AS SELECT dbms_lob.compare ( col1 ,col2 ,3 ,5 ,4 ) FROM emp a ,emp b ;
```

**Output**

```sql
CREATE UNLOGGED TABLE abc AS ( SELECT MIG_ORA_EXT.MIG_CLOB_COMPARE ( col1 ,col2 ,3 ,5 ,4 ) FROM emp a ,emp b ) ;
```

**Input - DBMS_LOB.COMpare in SQL used inside a function (NVL2)**

```sql
SELECT REPLACE( NVL2( DBMS_LOB.COMpare ( ENAME ,Last_name ) ,'NO NULL' ,'ONE NULL' ) ,'NULL' ) FROM emp ;
```

**Output**

---

**Data Warehouse Service Tool Guide**

Issue 03 (2019-08-08)

Copyright © Huawei Technologies Co., Ltd.
SELECT REPLACE( DECODE ( MIG_ORA_EXT.MIG_CLOB_COMPARE ( ENAME , Last_name ) , NULL , 'ONE NULL' , 'NO NULL' ) , 'NULL' , '' ) FROM emp ;

**COMPARE in PL/SQL**

**Input - DBMS_LOB.COMPAR E in PL/SQL**

```sql
DECLARE
    V_CLOB CLOB;
    V_TEXT VARCHAR(1000);
    V_COMPARE_RES INT;
BEGIN
    V_CLOB := TO_CLOB('abcddedf');
    V_TEXT := '123454';
    V_COMPARE_RES := DBMS_LOB.COMpare(V_CLOB, TO_CLOB(V_TEXT));
    DBMS_OUTPUT.PUT_LINE(V_COMPARE_RES);
END;
/
```

**Output**

```sql
DECLARE
    V_CLOB CLOB;
    V_TEXT VARCHAR(1000);
    V_COMPARE_RES INT;
BEGIN
    V_CLOB := CAST('abcddedf' AS CLOB);
    V_TEXT := '123454';
    V_COMPARE_RES := MIG_ORA_EXT.MIG_CLOB_COMPARE(V_CLOB, CAST(V_TEXT AS CLOB));
    DBMS_OUTPUT.PUT_LINE(V_COMPARE_RES);
END;
/
```

**DBMS_LOB.INSTR**

DBMS_LOB.INSTR function returns the matching position of the n\(^{th}\) occurrence of the pattern in the LOB, starting from the offset specified.

**Input - DBMS_LOB.INSTR in SQL**

```sql
SELECT expr1, ..., DBMS_LOB.INSTR(str, septr, 1, 5)
FROM tab1
WHERE ...;
```

**Output**

```sql
SELECT expr1, ..., INSTR(str, septr, 1, 5)
FROM tab1
WHERE ...;
```

**Input - DBMS_LOB.INSTR in PL/SQL**

```sql
BEGIN
    ... pos := DBMS_LOB.INSTR(str, septr, 1, i);
    ... END;
/
```

**Output**

```sql
BEGIN
    ... pos := INSTR(str, septr, 1, i);
    ... END;
/
```
DBMS_LOB.SUBSTR

Input - DBMS_LOB.SUBSTR when MigDbmsLob parameter value is true

If MigDbmsLob parameter value is true, then migration happens. and if the parameter's value is false then migration does not take place.

Input

select dbms_lob.substr('!2d3d4dd!',1,5);

Output

If the config param is true, it should be migrated as below:
select substr('!2d3d4dd!',5,1);

If false, it should be retained as it is:
select dbms_lob.substr('!2d3d4dd!',1,5);

Input

select dbms_lob.substr('!2d3d4dd!',5);

Output

If the config param is true, it should be migrated as below:
select substr('!2d3d4dd!',1,5);

If false, it should be retained as it is:
select dbms_lob.substr('!2d3d4dd!',5);

6.13.8.4 String Functions

This section contains information on the String Functions. It is organized as follows:

- LISTAGG
- STRAGG
- WM_CONCAT
- NVL2 and REPLACE
- QUOTE

LISTAGG

LISTAGG is used to order data within each group specified in the ORDER BY clause and concatenates the values of the measure column.

Figure 6-39 Input - Listagg

```
SELECT
depno,
ename
, LISTAGG ( 
  name 
, ' ' 
) OVER ( PARTITION BY depno , ename ORDER BY ename ) AS rn
FROM emp
ORDER BY
depno , ename
```
Figure 6-40 Output - Listagg

It depends on the following configuration item:

**MigSupportForListAgg=false**

**Input - LISTAGG**

```
SELECT LISTAGG(BRANCH_ID, ',') WITHIN GROUP(ORDER BY AREA_ORDER) PRODUCTRANGE
FROM (SELECT DISTINCT VB.BRANCH_ID,
  VB.VER_ID,
  VB.AREA_ORDER
FROM SPMS_VERSION_BRANCH VB, SPMS_NODE_SET NS
WHERE VB.BRANCH_TYPE IN ('1', '3')
  AND VB.AGENCY_BRANCH = NS.BRANCH_ID);
```

**Output**

```
SELECT LISTAGG (BRANCH_ID, ',') WITHIN GROUP (ORDER BY AREA_ORDER) PRODUCTRANGE
FROM ( SELECT
  DISTINCT VB.BRANCH_ID,
  VB.VER_ID,
  VB.AREA_ORDER
FROM SPMS_VERSION_BRANCH VB, SPMS_NODE_SET NS
WHERE VB.BRANCH_TYPE IN ('1', '3')
  AND VB.AGENCY_BRANCH = NS.BRANCH_ID);
```

**STRAGG**

*STRAGG* is a string aggregation function used to collect values from multiple rows into a single, comma delimited string.

Figure 6-41 Input - Stragg
Figure 6-42 Output - Stragg

SELECT DEPTNO, ENAME, STRAGG(ename) over (partition by deptno order by ename RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS ENAME_STR FROM EMP;

Input - STRAGG

SELECT DEPTNO, ENAME, STRING_AGG (ename, ',') over (partition BY deptno ORDER BY ename RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS ENAME_STR FROM EMP;

Output

WM_CONCAT

WM_CONCAT is used to aggregate data from a number of rows into a single row, giving a list of data associated with a specific value.

NVL2 and REPLACE

NVL2 (expression, value1, value2) is a function used to determine the value returned by a query based on whether a specified expression is null or not null. If expression is not null, then NVL2 returns value1. If expression is null, then NVL2 returns value2.
**Input - NVL2**

NVL2(Expr1, Expr2, Expr3)

**Output**

DECODE(Expr1, NULL, Expr3, Expr2)

The **REPLACE** function is used to return char with every occurrence of search_string replaced with replacement_string. If replacement_string is omitted or null, then all occurrences of search_string are removed.

In Oracle, the REPLACE function requires two parameters with a third parameter being optional. The REPLACE function in DWS requires three parameters.

**Input - Nested REPLACE**

```sql
CREATE OR REPLACE FUNCTION F_REPLACE_COMMA ( IS_STR IN VARCHAR2 ) RETURN VARCHAR2 IS BEGIN
  IF IS_STR IS NULL THEN RETURN NULL ;
  ELSE RETURN REPLACE( REPLACE( IS_STR ,'a' ) ,CHR ( 10 ) ) ;
  END IF ;
END F_REPLACE_COMMA ;
/
```

**Output**

```sql
CREATE OR REPLACE FUNCTION F_REPLACE_COMMA ( IS_STR IN VARCHAR2 ) RETURN VARCHAR2 IS BEGIN
  IF IS_STR IS NULL THEN RETURN NULL ;
  ELSE RETURN REPLACE( REPLACE( IS_STR ,'a' ,'' ) ,CHR ( 10 ) ,'' ) ;
  END IF ;
end ;
/
```

**Input - More than 1 REPLACE**

```sql
SELECT
  REPLACE( 'JACK and JUE' ,'J', '' ) "Changes"
,REPLACE( 'JACK1 and JUE' ,'J' ) "Changes1"
,REPLACE( 'JACK2 and JUE' ,'J' ) "Changes2"
FROM DUAL ;
```

**Output**

```sql
SELECT
  REPLACE( 'JACK and JUE' ,'J' ,'' ) "Changes"
,REPLACE( 'JACK1 and JUE' ,'J' ,'' ) "Changes1"
,REPLACE( 'JACK2 and JUE' ,'J' ,'' ) "Changes2"
FROM DUAL ;
```

**Input - REPLACE with 3 parameters**

```sql
SELECT
  REPLACE( '123tech123' ,'123', '1')
FROM dual ;
```
Output

```sql
SELECT
    REPLACE('123tech123', '123', '1')
FROM
dual
;
```

**QUOTE**

**QUOTE** allows the user to embed single-quotes in literal strings without having to resort to double quotes that are, a literal string can be specified without having to double up on the single quote characters.

For example:

```sql
SELECT q'[I'm using quote operator in SQL statement]' "Quote (q) Operator" FROM dual;
```

**Figure 6-45** Input - Quote

```sql
SELECT
    q'[It's a string quote operator.]
FROM
dual
;
```

**Figure 6-46** Output - Quote

```sql
SELECT
    $q$It's a string quote operator.$q$
FROM
dual
;
```

### 6.13.8.5 Analytical Functions

Analytical functions compute an aggregate value based on a group of rows. They differ from aggregate functions in that they return multiple rows for each group. Analytical functions are commonly used to compute cumulative, moving, centered, and reporting aggregates. Migration Tool supports analytical functions including the RATIO_TO_REPORT function.

**Input - Analytical Functions**

```sql
SELECT empno, ename, deptno,
    COUNT(*) OVER() AS cnt,
    AVG(DISTINCT empno) OVER (PARTITION BY deptno) AS cnt_dst
FROM emp
ORDER BY empno;
```

**Output**

```sql
WITH aggDistQuery1 AS ( 
    SELECT
        deptno,
        AVG ( 
            DISTINCT empno 
        ) aggDistAlias1 
    FROM emp 
)
```
RATIO_TO_REPORT

RATIO_TO_REPORT is an analytic function which returns the proportion of a value over the total set of values.

Input - RATIO_TO_REPORT

```sql
SELECT last_name, salary
  , RATIO_TO_REPORT(salary) OVER () AS rr
FROM employees
WHERE job_id = 'PU_CLERK';
```

Output

```sql
SELECT last_name, salary
  , salary / NULLIF( SUM (salary) OVER( ), 0 ) AS rr
FROM employees
WHERE job_id = 'PU_CLERK';
```

Input - RATIO_TO_REPORT with AGGREGATE column in SELECT

```sql
SELECT
  Ename
  , Deptno
  , Empno
  , SUM (salary)
  , RATIO_TO_REPORT ( COUNT( DISTINCT Salary )
    ) OVER( PARTITION BY Deptno ) RATIO
FROM emp1
ORDER BY
  Ename
  , Deptno
  , Empno
;
```

Output

```sql
SELECT
  Ename
  , Deptno
  , Empno
  , SUM (salary)
  , COUNT( DISTINCT Salary ) / NULLIF( SUM ( COUNT( DISTINCT Salary ) ) OVER( PARTITION BY Deptno ), 0 ) RATIO
FROM
```
Input - `RATIO_TO_REPORT` with AGGREGATE column using extending grouping feature but `RATIO TO REPORT` column `COUNT( Salary )` is not present in select columns list

Use the `extendedGroupByClause` configuration parameter to configure migration of the extended GROUP BY clause.

```sql
SELECT
    Ename,
    Deptno,
    Empno,
    SUM(salary) Over( Partition By Deptno ) RATIO
FROM
    emp1
GROUP BY
    GROUPING SETS (Ename,Deptno,Empno)
ORDER BY
    Ename,Deptno,Empno;
```

Output

```sql
SELECT
    Ename,
    Deptno,
    Empno,
    ColumnAlias1,
    aggColumnalias1 / NULLIF(SUM(aggColumnalias1) Over( Partition By Deptno ) ,0) RATIO
FROM
    (SELECT
         SUM(salary) AS ColumnAlias1
        ,COUNT( Salary ) aggColumnalias1
        ,NULL AS Deptno
        ,NULL AS Empno
        ,Ename
    FROM
        emp1
    GROUP BY
        Ename)
        UNION
UNION
    ALL SELECT
         SUM(salary) AS ColumnAlias1
        ,COUNT( Salary ) aggColumnalias1
        ,Deptno
        ,NULL AS Empno
        ,NULL AS Ename
    FROM
        emp1
    GROUP BY
        Deptno
    UNION
UNION
    ALL SELECT
```
6.13.8.6 Regular Expression Functions

Regular expressions specify patterns to search for in string data using standardized syntax conventions. In Oracle, regular expressions are implemented using a set of SQL functions that allow you to search and manipulate string data.

Migration Tools supports `REGEXP_INSTR`, `REGEXP_SUBSTR` and `REGEXP_REPLACE` regular expressions as follows:

- Regexp (REGEXP_INSTR and REGEXP_SUBSTR) that include the `sub_expr` parameter are not supported. If the input script includes `sub_expr`, the migration tool will log an error for it.
- Regexp (REGEXP_INSTR, REGEXP_SUBSTR and REGEXP_REPLACE) uses the `match_param` parameter to set the default matching behavior. The migration tool supports the values of "i" for case-insensitive matching and "c" for case-sensitive matching for this parameter. Other values for `match_param` are not supported.
- Regexp (REGEXP_INSTR) uses the `return_option` parameter to set what is returned for regexp matches. The migration tool supports the value 0 (zero) for this parameter. Other values for `return_option` are not supported.

**REGEXP_INSTR**

REGEXP_INSTR extends the functionality of the INSTR function by supporting regular expression pattern for the search string. REGEXP_INSTR with 2 to 6 parameters are supported for migration.

The `sub_expr` parameter (parameter #7) is available in Oracle but is not supported for migration. If the input script includes `sub_expr`, the migration tool will log an error for it.

For `return_option`, the value 0 (zero) is supported. Other values for `return_option` are not supported.

For `match_param`, values of "i" for case-insensitive matching and "c" for case-sensitive matching are supported. Other values for `match_param` are not supported.

```sql
REGEXP_INSTR( string, pattern, [start_position], [nth_appearance], [return_option], [match_param] )
```
Bulk Operations

- **Input - REGEXP_INSTR**

```
SELECT REGEXP_INSTR('TechOnTheNet is a great resource', 't')
FROM dual;
```

**Output**

```
SELECT MIG_ORA_EXT.REGEXP_INSTR('TechOnTheNet is a great resource', 't')
FROM dual;
```

- **Input - REGEXP_INSTR with 7 arguments (Invalid)**

```
SELECT Empno, ename,
   REGEXP_INSTR(ename, 'a|e|i|o|u', 1, 1, 0, 'i', 7) AS Dname
FROM emp19;
```

**Output**

In this case the input expression has 7 arguments. Since the tool supports REGEXP_INSTR with 2 to 6 arguments, an error will also be logged. "Seven(7) arguments for REGEXP_INSTR function are not supported."

```
SELECT Empno, ename,
   REGEXP_INSTR(ename, 'a|e|i|o|u', 1, 1, 0, 'i', 7) AS Dname
FROM emp19;
```

BLogic Operations

- **Input - REGEXP_INSTR**

```
CREATE OR REPLACE FUNCTION myfct
RETURN VARCHAR2
IS
  res VARCHAR2(200) ;
BEGIN
  res := 100 ;
  INSERT INTO emp19 RW(RW.empno,RW.ename,dname) SELECT res, RWN.ename key,
         regexp_instr(ename ,'[ae]',4,2,0, 'i')   as Dname FROM   emp19 RWN ;
  RETURN res ;
END ;
/
```

**Output**

```
CREATE OR REPLACE FUNCTION myfct RETURN VARCHAR2 IS res VARCHAR2 ( 200 ) ;
BEGIN
  res := 100 ;
```
REGEXP_SUBSTR

REGEXP_SUBSTR extends the functionality of the SUBSTR function by supporting regular expression pattern for the search string. REGEXP_SUBSTR with 2 to 5 parameters are supported for migration.

The sub_expr parameter (parameter #6) is available in Oracle but is not supported for migration. If the input script includes sub_expr, the migration tool will log an error for it.

For match_param, values of "i" for case-insensitive matching and "c" for case-sensitive matching are supported. Other values for match_param are not supported.

REGEXP_SUBSTR(
    string,
    pattern,
    [start_position,]
    [nth_appearance,]
    [match_param,]
    [sub_expr]
)

Bulk Operations

- Input - REGEXP_SUBSTR

SELECT
    Ename,
    REGEXP_SUBSTR( 'Programming', '(\w).*?\1' ,1 ,1 , 'i' )
FROM
    emp16
;

Output

SELECT
    Ename,
    MIG_ORA_EXT.REGEXP_SUBSTR ( 'Programming' , '(\w).*?\1' ,1 ,1 , 'i' )
FROM
    emp16
;

- Input - REGEXP_SUBSTR

SELECT
    REGEXP_SUBSTR( '1234567890' , '(123)(4(56)(78))' ,1 ,1 , 'i' )
FROM
    DUAL
;

Output

SELECT
    MIG_ORA_EXT.REGEXP_SUBSTR (
REGEXP_REPLACE

REGEXP_REPLACE extends the functionality of the REPLACE function by supporting regular expression pattern for the search string. REGEXP_REPLACE with 2 to 6 parameters are supported for migration.
For `match_param`, values of "i" for case-insensitive matching and "c" for case-sensitive matching are supported. Other values for `match_param` are not supported.

```sql
REGEXP_REPLACE(
    string,
    pattern,
    [replacement_string,]
    [start_position,]
    [nth_appearance,]
    [match_param]
)
```

### Bulk Operations

#### Input - `REGEXP_REPLACE`

```sql
SELECT
    testcol,
    regexp_replace( testcol, '\([[:digit:]]\{3\}\).\([[:digit:]]\{3\}\).\([[:digit:]]\{4\}\)' , '\1 \2-\3' )
FROM
    test
WHERE
    LENGTH( testcol ) = 12
;
```

#### Output

```sql
SELECT
    testcol,
    MIG_ORA_EXT.REGEXP_REPLACE ( testcol, '\([[:digit:]]\{3\}\).\([[:digit:]]\{3\}\).\([[:digit:]]\{4\}\)' , '\1 \2-\3' )
FROM
    test
WHERE
    LENGTH( testcol ) = 12
;
```

#### Input - `REGEXP_REPLACE`

```sql
SELECT
    UPPER( regexp_replace ( 'foobarbequebazilbarfbonk barbeque' , '(b[^b]+)(b[^b]+)' ) )
FROM
    DUAL
;
```

#### Output

```sql
SELECT
    UPPER( MIG_ORA_EXT.REGEXP_REPLACE ( 'foobarbequebazilbarfbonk barbeque' , '(b[^b]+)(b[^b]+)' ) )
FROM
    DUAL
;
```

#### Input - `REGEXP_REPLACE` with 7 arguments (Invalid)

```sql
SELECT
    REGEXP_REPLACE( 'TechOnTheNet', 'a|e|i|o|u' , 'Z' , 1 , 1 , 'i' , '\(1 \2-\3\)' ) AS First_Occurrence
FROM
    emp
;
```

#### Output

In this case the input expression has 7 arguments. Since the tool supports `REGEXP_REPLACE` with 2 to 6 arguments, an error will also be logged. "Too many arguments for `REGEXP_REPLACE` function [Max:6 argument(s) is/are allowed]."
SELECT
    REGEXP_REPLACE( 'TechOnTheNet', 'a|e|i|o|u', 'Z', 1, 1, 'i', '(\1) \2-\3' ) AS First_Occurrence
FROM
    emp
;

BLogic Operations

- Input - REGEXP_REPLACE

CREATE OR REPLACE FUNCTION myfct
RETURN VARCHAR2
IS
res VARCHAR2(200) 
BEGIN
    res := 100;
    INSERT INTO emp19 (empno, ename, dname) SELECT res, RWN.ename "KEY", MIG_ORA_EXT.REGEXP_REPLACE ( 'TechOnTheNet', 'a|e|i|o|u', 'Z', 1, 1, 'i' ) as Dname FROM emp19 RWN;
    RETURN res;
END;
/

Output

CREATE OR REPLACE FUNCTION myfct RETURN VARCHAR2 IS res VARCHAR2 ( 200 )
BEGIN
    res := 100;
    INSERT INTO emp19 ( empno ,ename ,dname ) SELECT res ,RWN.ename "KEY" ,MIG_ORA_EXT.REGEXP_REPLACE ( 'TechOnTheNet' , 'a|e|i|o|u' , 'Z' , 1 , 1 , 'i' ) as Dname FROM emp19 RWN;
    RETURN res;
END;
/

LISTAGG/regexp_replace/regexp_instr

It depends on the following configuration items:

- MigSupportForListAgg=false
- MigSupportForRegexReplace=false

Input- REMOVE LISTAGG/regexp_replace/regexp_instr

SELECT LISTAGG(T.OS_SOFTASSETS_ID,',' ) WITHIN GROUP(ORDER BY T.SOFTASSETS_ID)
    INTO V_OS_SOFTASSETS_IDS
    FROM SPMS_SYSSOFT_PROP_APPR T
    WHERE T/APPR_ID = I_APPR_ID
    AND T/SYSSOFT_PROP = '001';
    V_ONLY_FILE_NAME := REGEXP_REPLACE (I_FILENAME, '.*/', '');

THEN v_auth_type := 102;
ELSIF v_status IN ('0100', '0200')
    AND REGEXP_INSTR (v_role_str, ',(411|414),') > 0

Output

"SELECT LISTAGG(T.OS_SOFTASSETS_ID,',' ) WITHIN GROUP(ORDER BY T.SOFTASSETS_ID)
    INTO V_OS_SOFTASSETS_IDS
    FROM SPMS_SYSSOFT_PROP_APPR T
    WHERE T/APPR_ID = I_APPR_ID
    AND T/SYSSOFT_PROP = '001';
    V_ONLY_FILE_NAME := REGEXP_REPLACE (I_FILENAME, ';','');
6.13.8.7 Other Functions

This section describes the following functions:

- **Type Casting**
- **User Environment**

**Type Casting**

The Oracle/PLSQL CAST function converts one data type to another.

**CAST (AS TEXT)**

Static values are replaced with `CAST AS TEXT` which is old implementation but for PL/SQL procedures and functions when static values are used in the `SELECT INTO` clause it does not migrate, input is same as output.

**Input - CAST (AS TEXT)**

```sql
BEGIN
  IF V_DETAILSTATUS = '002'
  THEN
    meta.p_mt_std_cover_log.r_cover_log ( '384969028' , '566' , '35' ) ;
    SELECT 'abc' INTO V_RECL_USER
    FROM STAT_DETAIL
    WHERE DAPCODE = V_DAPCODE
    AND DETAIL_ID = V_DETAIL_ID ;
  ELSEIF ... END;
```

**Output**

```sql
BEGIN
  IF V_DETAILSTATUS = '002'
  THEN
    meta.p_mt_std_cover_log.r_cover_log ( '384969028' , '566' , '35' ) ;
    SELECT 'abc' INTO V_RECL_USER
    FROM STAT_DETAIL
    WHERE DAPCODE = V_DAPCODE
    AND DETAIL_ID = V_DETAIL_ID ;
  ELSEIF ... END;
```

**TO_CLOB**

In Oracle, the CLOB data type stores variable-length character data (character large objects) in the database character set that can be single-byte or multi-byte (supports more than 4 GB). The TO_CLOB function is used to convert a string (CHAR, VARCHAR2, NCHAR, NVARCHAR2, CLOB, or NCLOB) to the CLOB data type. Migration Tool supports TO_CLOB in SQL and PL/SQL.
Input - TO_CLOB in SQL

```sql
create table clob_t ( c1 int, c2 clob );
insert into clob_t values ( 1, TO_CLOB('abcddedf') );
```

Output

```sql
create table clob_t ( c1 int, c2 clob );
insert into clob_t values ( 1, CAST('abcddedf' AS CLOB) );
```

Input - TO_CLOB in PL/SQL

```sql
declare v_clob clob;
v_text varchar(1000);
begi  in Sn
    v_clob := TO_CLOB('abcddedf');
    insert into clob_t values ( 2, v_clob );

    v_text := '123454';
    insert into clob_t values ( 3, TO_CLOB(v_text || v_clob) );
end;
/
```

Output

```sql
declare v_clob clob;
v_text varchar(1000);
begin
    v_clob := CAST('abcddedf' AS CLOB);
    insert into clob_t values ( 2, v_clob );

    v_text := '123454';
    insert into clob_t values ( 3, CAST(v_text || v_clob AS CLOB) );
end;
/
```

sys_guid( )

Table column name appeared with default value as function name not supported like - SYS_ID VARCHAR2 (32) DEFAULT sys_guid( )

Input

```sql
CREATE TABLE SAD.FND_DATA_CHANGE_LOGS_T
(    LOGID NUMBER,
    TABLE_NAME VARCHAR2(40) NOT NULL ENABLE,
    TABLE_KEY_COLUMNS VARCHAR2(200),
    TABLE_KEY_VALUES VARCHAR2(200),
    COLUMN_NAME VARCHAR2(40) NOT NULL ENABLE,
    COLUMN_CHANGE_FROM_VALUE VARCHAR2(200),
    COLUMN_CHANGE_TO_VALUE VARCHAR2(200),
    DESCRIPTION VARCHAR2(500),
    SYS_ID VARCHAR2(32) DEFAULT sys_guid()
);
```

Output

```sql
CREATE TABLE sad.fnd_data_change_logs_t
(    logid                                  NUMBER,
    ,table_name                        VARCHAR2 (40) NOT NULL /* ENABLE */
    ,table_key_columns              VARCHAR2 (200),
    ,table_key_values                VARCHAR2 (200),
    ,column_name                     VARCHAR2 (40) NOT NULL /* ENABLE */
    ,column_change_from_value   VARCHAR2 (200),
    ,column_change_to_value      VARCHAR2 (200),
    ,description                         VARCHAR2 (500)
);
```
sys.dbms_job.submit

The functions call with dbms_job using sys is not supported that is sys.dbms_job.submit (job => v_jobid, what => 'begin eip_htm_integration_pkg.import_instruction_job; end;', next_date => SYSDATE);

If you remove sys from above call, then it is working that is dbms_job.submit (job => v_jobid, what => 'begin eip_htm_integration_pkg.import_instruction_job; end;', next_date => SYSDATE);

SCRIPTS : EIP_HTM_INTEGRATION_PKG.sql,
SAD_CALC_CUSTOM_DRAWBACK_PKG.sql

Input

CREATE OR REPLACE PACKAGE BODY "SAD"."EIP_HTM_INTEGRATION_PKG" IS
PROCEDURE greate_import_instruction_job IS
  v_jobid NUMBER;
BEGIN
  IF bas_lookup_misc_pkg.exits_run_job('eip_htm_integration_pkg.import_instruction_job') = 'N'
  THEN
    sys.dbms_job.submit (job => v_jobid,
                          what => 'begin eip_htm_integration_pkg.import_instruction_job;
                                  end;',
                          next_date => SYSDATE);
    COMMIT;
  END IF;
  ---
  END greate_import_instruction_job;
  END EIP_HTM_INTEGRATION_PKG;

Output

CREATE OR REPLACE PROCEDURE SAD.EIP_HTM_INTEGRATION_PKG#greate_import_instruction_job IS v_jobid NUMBER ;
ex_data_error EXCEPTION ;
ex_prog_error EXCEPTION ;
BEGIN
  IF BAS_LOOKUP_MISC_PKG#exits_run_job ( 'eip_htm_integration_pkg.import_instruction_job' ) = 'N' THEN
    sys.dbms_job.submit ( job => v_jobid ,what => 'begin eip_htm_integration_pkg.import_instruction_job; end;' ,next_date => SYSDATE ) ;
/* COMMIT; */
NULL ;
END IF ;
---
END ;

Input

CREATE OR REPLACE PROCEDURE greate_import_instruction_job IS
  v_jobid NUMBER;
BEGIN
  IF bas_lookup_misc_pkg.exits_run_job('eip_htm_integration_pkg.import_instruction_job') = 'N'
  THEN
    sys.dbms_job.submit (job => v_jobid,
                          what => 'begin eip_htm_integration_pkg.import_instruction_job;
                                  end;',
                          next_date => SYSDATE);
    COMMIT;
  END IF;
  ---
  END greate_import_instruction_job;
  END EIP_HTM_INTEGRATION_PKG;
END IF;
END greate_import_instruction_job;
/

**Output**

CREATE OR REPLACE PROCEDURE greate_import_instruction_job
PACKAGE
IS
v_jobid NUMBER;
BEGIN
  IF bas_lookup_misc_pkg.exits_run_job ( 'eip_htm_integration_pkg.import_instruction_job' ) = 'N' THEN
    dbms_job.submit ( job => v_jobid ,what => 'begin
eip_htm_integration_pkg.import_instruction_job; end;' ,next_date => SYSDATE ) ;
/* COMMIT; */
  NULL ;
END IF;
END ;
/

**User Environment**

In Oracle, USERENV is a namespace that describes the current session, and it is used to obtain information about the current Oracle session. Migration tool supports the migration of USERENV statements as shown in the examples.

The following scenarios are not supported and Migration Tool will log an exception for them:

<table>
<thead>
<tr>
<th>Error Cause</th>
<th>Sample Input</th>
<th>Sample Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsupported Parameter</td>
<td>SELECT userenv('LANG') LANG FROM DUAL;</td>
<td>SELECT userenv('LANG') LANG FROM DUAL;</td>
</tr>
<tr>
<td>Parameter is a variable</td>
<td>SELECT userenv(ENTRYID) LANG FROM DUAL;</td>
<td>SELECT userenv(ENTRYID) LANG FROM DUAL;</td>
</tr>
<tr>
<td>No parameter provided</td>
<td>SELECT userenv() LANG FROM DUAL;</td>
<td>SELECT userenv() LANG FROM DUAL;</td>
</tr>
<tr>
<td>Empty parameter provided</td>
<td>SELECT userenv('SID','INSTANCE' ) LANG FROM DUAL;</td>
<td>SELECT userenv('SID','INSTANCE' ) LANG FROM DUAL;</td>
</tr>
<tr>
<td>More than one parameter provided</td>
<td>SELECT userenv(&quot; &quot;) LANG FROM DUAL;</td>
<td>SELECT userenv(&quot; &quot;) LANG FROM DUAL;</td>
</tr>
<tr>
<td>One valid syntax and one invalid syntax (unsupported parameter)</td>
<td>SELECT USERENV( 'sid' ) ,USERENV( 'lang' ) FROM dual ;</td>
<td>SELECT USERENV( 'sid' ) ,USERENV( 'lang' ) FROM dual ;</td>
</tr>
<tr>
<td>Double quotes in input</td>
<td>SELECT USERENV( &quot;sid&quot; ) FROM dual ;</td>
<td>SELECT USERENV( &quot;sid&quot; ) FROM dual ;</td>
</tr>
</tbody>
</table>
Before using this function, perform the following operations:

1. Create and use the MIG_ORA_EXT schema.
2. Copy the contents of the custom script and execute the script in all target databases for which migration is to be performed. For details, see Custom DB Script Configuration and Migration Process.

### Input - USERENV

Input - USERENV

```
SELECT userenv('SID') SID FROM DUAL;
```

Output

```
SELECT mig_ora_ext.userenv('SID') SID FROM DUAL;
```

### Input - USERENV

Input - USERENV

```
CREATE OR REPLACE PROCEDURE test_userEnvFun AS
  v_sid NUMBER (8) := 0;
  v_instid NUMBER (8) := 0;
BEGIN
  SELECT USERENV('sid'),USERENV('instance') INTO v_sid,v_instid
  FROM dual;
end test_userEnvFun;
/
```

Output

```
CREATE OR REPLACE PROCEDURE test_userEnvFun AS
  v_sid NUMBER (8) := 0;
  v_instid NUMBER (8) := 0;
BEGIN
  SELECT MIG_ORA_EXT.USERENV('sid'),MIG_ORA_EXT.USERENV('instance') INTO v_sid,v_instid
  FROM dual;
end;
/
```

### 6.13.9 PL/SQL

#### 6.13.9.1 Overview

This section contains the migration syntax for migrating Oracle PL/SQL. The migration syntax decides how the supported keywords/features are migrated.

PL/SQL is a combination of SQL along with the procedural features of programming languages.

#### 6.13.9.2 EDITIONABLE

EDITIONABLE keyword is not supported in Gauss. Hence should be removed in target database.

Input – EDITIONABLE
CREATE OR REPLACE EDITIONABLE PACKAGE "PACK1"."PACKAGE_SEND_MESSAGE"
AS
  TYPE filelist IS REF CURSOR;
  PROCEDURE get_message_info (in_userid          IN    VARCHAR2,
                              in_branchid        IN    VARCHAR2,
                              in_appverid        IN    VARCHAR2,
                              in_app_list_flag   IN    VARCHAR2,
                              in_filetype        IN    VARCHAR2,
                              in_filestate       IN    VARCHAR2,
                              o_retcode          OUT   VARCHAR2,
                              o_errormsg         OUT   VARCHAR2,
                              o_seq              OUT   VARCHAR2,
                              o_totalnum         OUT   NUMBER,
                              o_filelist         OUT   filelist);

Output
/*~~PACKAGE_SEND_MESSAGE~~*/
CREATE
  SCHEMA PACKAGE_SEND_MESSAGE
;

6.13.9.3 Variable Assignment

Figure 6-47 Input - PL/SQL

BEGIN
...
  vRowCount := SQL%ROWCOUNT;
...
END;

Figure 6-48 Output - PL/SQL

BEGIN
...
  vRowCount := SQL%ROWCOUNT;
...
END;

6.13.9.4 END

END with label is not supported in GaussDB 100. So, the label name is removed during migration.

Input - END with procedure name

CREATE OR REPLACE PROCEDURE sp_ins_emp
...
...
END sp_ins_emp;

Output

CREATE OR REPLACE PROCEDURE sp_ins_emp
...
...
END;
### 6.13.9.5 EXCEPTION Handling

DWS V100R200C60 do not support EXCEPTION handling. To migrate scripts to V100R200C60, the `exceptionHandler` parameter must be set to `True`.

For Migration Tool 18.2.0, this parameter value must be the default `False`.

**Figure 6-49 Input - EXCEPTION Handling**

```sql
CREATE OR REPLACE FUNCTION get_salary (n_emp_id NUMBER) RETURN NUMBER
IS
    n_salary NUMBER (8,2)
    emp employees;
BEGIN
    SELECT
        salary INTO n_salary
    FROM
        employees
    WHERE
        id = n_emp_id;
    RETURN n_salary;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN RETURN NULL;
        WHEN TOO_MANY_ROWS THEN RETURN NULL;
END;
```

**Figure 6-50 Output - EXCEPTION Handling**

### 6.13.9.6 Subtransaction Handling

This parameter must be `True`, which is the default value.
6.13.9.7 STRING

This is handled by using VARCHAR.

Figure 6-53 Input - STRING
LONG 

LONG datatype is migrated as TEXT.

**Input - LONG**

```sql
CREATE OR REPLACE FUNCTION fn_proj_det
( i_proj_cd INT )
RETURN LONG
IS
  v_proj_det LONG;
BEGIN
  SELECT proj_det INTO v_proj_det FROM project WHERE proj_cd = i_proj_cd;
  RETURN v_proj_det;
END;
/
```

**Output**

```sql
CREATE OR REPLACE FUNCTION fn_proj_det
( i_proj_cd INT )
RETURN TEXT
IS
  v_proj_det TEXT;
BEGIN
  SELECT proj_det INTO v_proj_det FROM project WHERE proj_cd = i_proj_cd;
  RETURN v_proj_det;
END;
/
```

**6.13.9.9 RESULT_CACHE**

When a function with result cache is called, Oracle executes the function, adds the result to the result cache and then returns the function.

When the function call is repeated, Oracle fetches the results from the cache rather than re-execute the function.

Under certain scenarios, this caching behavior can result in significant performance gains.

This keyword is not supported in the target database and is removed from the target file.
6.13.9.10 Relational Operator with Space

The relational operators (<=, >=, !=) that have a space in them are not supported in DWS. The Migration Tool removes the space between the operators.
### 6.13.9.11 Substitution Variables

Substitution variables is a feature of Oracle SQL*Plus tool. When a substitution variable is used in a statement, SQL*Plus requests an input value and rewrites the statement to include it. The rewritten statement is passed to the Oracle database. When the input Oracle scripts contain any substitution variable, the migration tool displays the following message. The message is logged on the console and in the log file.

```plaintext
**************************************************************************
USER ATTENTION!!! Variable: &bbid should be substituted in the file:
/home/testmigration/V100R002C60/MigrationTool/Input/proc_frss_jczbsc.sql
Variable: &wdbs should be substituted in the file:
/home/testmigration/V100R002C60/MigrationTool/Input/proc_frss_jczbsc.sql
Variable: &batch_no should be substituted in the file:
/home/testmigration/V100R002C60/MigrationTool/Input/proc_frss_jczbsc.sql
**************************************************************************
```

### 6.13.9.12 PARALLEL_ENABLE

In Oracle, PARALLEL_ENABLE is used to enable partitioning of the workload by using parallel execution of a function.

**Input - PARALLEL_ENABLE**

```sql
CREATE OR REPLACE FUNCTION F_REPLACE_COMMA (IS_STR IN VARCHAR2)
RETURN VARCHAR2
parallel_enable
IS
BEGIN
  IF IS_STR IS NULL THEN
    RETURN NULL;
  ELSE
    RETURN REPLACE(REPLACE(IS_STR, CHR(13) || CHR(10), ''), ',', ', ');
  END IF;
END F_REPLACE_COMMA;
/
```

**Output**

```sql
CREATE OR REPLACE FUNCTION F_REPLACE_COMMA (IS_STR IN VARCHAR2)
RETURN VARCHAR2
IS
BEGIN
  IF IS_STR IS NULL THEN
    RETURN NULL;
  ELSE
    RETURN REPLACE(REPLACE(IS_STR, CHR(13) || CHR(10), '), ', ', ');
  END IF;
END;
/
```
6.13.9.13 TRUNCATE TABLE

The TRUNCATE TABLE statement in Oracle is used to remove all records from a table. It performs the same function as a DELETE statement without a WHERE clause. After truncating, the table will exist but it will be empty. Migration Tool supports migration of TRUNCATE TABLE statements with static table names only. Migration of TRUNCATE TABLE statements with dynamic table names is not supported by the tool.

**NOTE**

The tool does not support migration of TRUNCATE TABLE statements with dynamic table names.

Example: l_table := 'truncate table ' || itable_name

In this example, itable_name indicates a dynamic table name and is not supported by the migration tool. The unsupported statements will be copied verbatim in the migrated scripts.

**Input - TRUNCATE TABLE with Execute Immediate**

CREATE OR REPLACE PROCEDURE schema1.proc1 AS
BEGIN
    EXECUTE IMMEDIATE 'TRUNCATE TABLE QUERY_TABLE';
End proc1;
/

**Output**

CREATE OR REPLACE PROCEDURE schema1.proc1 AS BEGIN
    EXECUTE IMMEDIATE 'TRUNCATE TABLE schema1.QUERY_TABLE';
end ;
/

**Input - TRUNCATE TABLE inside procedure**

**NOTE**

Migration tool does not add schema names for dynamic PL/SQL statements.

CREATE OR REPLACE PROCEDURE schemName.sp_dd_table ( itable_name VARCHAR2 ) IS l_table VARCHAR2 ( 255 ) ;
BEGIN
    l_table := 'truncate table ' || itable_name ;
    ---- dbms_utility.exec_ddl_statement(l_table);
dbms_output.put_line ( itable_name || ' Truncated' ) ;
END sp_dd_table ;
/

**Output**

CREATE OR REPLACE PROCEDURE schemName.sp_dd_table ( itable_name VARCHAR2 ) IS l_table VARCHAR2 ( 255 ) ;
BEGIN
    l_table := 'truncate table ' || itable_name ;
    /*
     * dbms_utility.exec_ddl_statement(l_table); */
dbms_output.put_line ( itable_name || ' Truncated' ) ;
end ;
/
6.13.9.14 ALTER SESSION

The **ALTER SESSION** statement in Oracle is used to set or modify the parameters and behavior of the database connection. The statement stays in effect until you disconnect from the database. The Migration Tool supports the migration of ALTER SESSION as follows:

- ALTER SESSION with ADVISE, ENABLE, DISABLE, CLOSE and FORCE clauses are migrated as commented scripts.
- ALTER SESSION with SET CLAUSE parameter (Example: NLS_DATE_FORMAT, NLS_DATE_LANGUAGE, and so on) are copied verbatim.

**NOTE**

The tool does not support migration of ALTER SESSION statements that have a variable for the command clause.

Example: EXECUTE IMMEDIATE 'alter session ' || *command_val* || 'parallel ' || *type_value*.

In this example, *command_val* is a variable and this is not supported by the migration tool. The unsupported statements will be copied verbatim in the migrated scripts.

**Input - ALTER SESSION**

```
ALTER SESSION ENABLE PARALLEL DDL;
ALTER SESSION ADVISE COMMIT;
ALTER SESSION CLOSE DATABASE LINK local;
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY MM DD HH24:MI:SS';
ALTER SESSION SET current_schema = 'isfc';
```

**Output**

```
/*ALTER SESSION ENABLE PARALLEL DDL;*/
/*ALTER SESSION ADVISE COMMIT;*/
/*ALTER SESSION CLOSE DATABASE LINK local;*/
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY MM DD HH24:MI:SS';
ALTER SESSION SET current_schema = 'isfc';
```

**Input - ALTER SESSION**

```
PROCEDURE PUBLIC .TEST_CALL is
command_val varchar2 ( 1000 ) ;
type_value number ;
BEGIN
    command_val := 'enable parallel ddl' ;
dbms_output.put_line ( mike ) ;
    -- execute immediate 'ALTER SESSION DISABLE GUARD' ;
    execute immediate 'ALTER SESSION ADVISE ROLLBACK' ;
    EXECUTE IMMEDIATE ' alter session ' || command_val || 'parallel ' || type_value ;
END TEST_CALL;
/
```

**Output**

```
CREATE OR REPLACE
PROCEDURE PUBLIC .TEST_CALL is
command_val varchar2 ( 1000 ) ;
type_value number ;
BEGIN
    command_val := 'enable parallel ddl' ;
dbms_output.put_line ( mike ) ;
    /* execute immediate 'ALTER SESSION DISABLE GUARD' ;*/
    execute immediate '/**ALTER SESSION ADVISE ROLLBACK*/' ;
    EXECUTE IMMEDIATE 'alter session ' || command_val || 'parallel ' || type_value;
END ;
/
```
### 6.13.9.15 AUTONOMOUS

**Input - AUTONOMOUS**

```sql
CREATE OR REPLACE EDITIONABLE PACKAGE BODY "Pack1"."DEMO_PROC" IS
    PROCEDURE log(proc_name IN VARCHAR2, info IN VARCHAR2) IS
        PRAGMA AUTONOMOUS_TRANSACTION;
END;
```

**Output**

```sql
CREATE OR REPLACE PROCEDURE DEMO_PROC.log ( proc_name IN VARCHAR2, info IN VARCHAR2 ) IS
    /*PRAGMA AUTONOMOUS_TRANSACTION;*/
END;
```

### 6.13.9.16 Procedure Call

**Procedure With No Parameter**

Procedure with no parameter.

Need to put () after procedure name while calling the same.

**Ex:** `pkg_etl.clear_temp_tables()`

**Input**

```sql
CREATE OR REPLACE PACKAGE BODY IC_STAGE.pkg_etl
AS
    PROCEDURE clear_temp_tables
    IS
        BEGIN
        NULL;
    END clear_temp_tables;
    END pkg_etl;
/;
CREATE OR REPLACE PACKAGE BODY IC_STAGE.PKG_REVN_ARPU
AS
    PROCEDURE AGGR_X_AGG00_REVN_DEALER (p_date PLS_INTEGER, p_days PLS_INTEGER)
    AS
        v_start_date   PLS_INTEGER;
        v_curr_date    PLS_INTEGER;
    BEGIN
        v_start_date := TO_CHAR (TO_DATE (p_date, 'yyyymmdd') - (p_days - 1), 'yyyymmdd');
        v_curr_date := p_date;
        WHILE (v_curr_date >= v_start_date)
        LOOP
            pkg_etl.clear_temp_tables;
            pkg_dw.bind_variable ('v_curr_date', v_curr_date);
            v_curr_date := TO_CHAR (TO_DATE (v_curr_date, 'yyyymmdd') - 1, 'yyyymmdd');
        END LOOP;
    END;
    END PKG_REVN_ARPU;
/;
```

**Output**

```sql
CREATE OR REPLACE PROCEDURE IC_STAGE.pkg_etl#clear_temp_tables PACKAGE IS
    BEGIN
    NULL;
    END;
/
CREATE OR REPLACE PROCEDURE IC_STAGE.PKG_REVN_ARPU#AGGR_X_AGG00_REVN_DEALER
(p_date INTEGER, p_days INTEGER)
```

---

**Data Warehouse Service**

**Tool Guide**

**Issue 03 (2019-08-08)**

Copyright © Huawei Technologies Co., Ltd.
PACKAGE AS

BEGIN

v_start_date := TO_CHAR( TO_DATE( p_date ,'yyyymmdd' ) - ( p_days - 1 ), 'yyyymmdd' ) ;

v_curr_date := v_start_date ;

WHILE ( v_curr_date >= v_start_date ) LOOP

pkg_etl#clear_temp_tables () ;

pkg_dwbind_variable ( 'v_curr_date' ,v_curr_date ) ;

v_curr_date := TO_CHAR( TO_DATE( v_curr_date ,'yyyymmdd' ) - 1,'yyyymmdd' ) ;

END LOOP ;

END ;

/

Function Name Having No Parameter Is Called

Function name having no parameter is called by function name with parameter is not supported in EXCEPTION statement that is SAD.SAD_CALC_ITEM_PKG_TEST_OB#error_msg ( ) but this function error_msg is defined without parameter.

CREATE OR REPLACE FUNCTION SAD.SAD_CALC_ITEM_PKG_TEST_OB#func_name RETURN VARCHAR2 IS MIG_PV_VAL_DUMMY_G_FUNC_NAME VARCHAR2 ( 30 ) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE ( current_schema ( ) --- BEGIN --- RETURN l_func_name ; END ;

SCRIPTS : SAD_CALC_ITEM_PKG_TEST_OB.sql, SAD_CALC_ITEM_PRI_TEST_OB.sql

Input

CREATE OR REPLACE PACKAGE BODY "SAD"."SAD_CALC_ITEM_PKG_TEST_OB" IS

PROCEDURE back_sad_cost_line_t(pi_contract_number IN VARCHAR2,
pi_quotation_id IN NUMBER,
pi_product_code IN VARCHAR2,
pi_process_batch_number IN NUMBER,
po_error_msg OUT VARCHAR2) IS

BEGIN

--- LOOP

INSERT INTO sad_cost_line_bak (processing_batch_number,
contract_number,
product_code,
quotation_id,
item_code,
refresh_date,
split_date,
error_msg,
created_by,
creation_date,
last_updated_by,
last_update_date)

Issue 03 (2019-08-08) Copyright © Huawei Technologies Co., Ltd. 541
VALUES 
(pi_process_batch_number,
cur_1.contract_number,
cur_1.product_code,
cur_1.quotation_id,
cur_1.item_code,
cur_1.refresh_date,
cur_1.split_date,
cur_1.error_msg,
cur_1.created_by,
cur_1.creation_date,
cur_1.last_updated_by,
cur_1.last_update_date);
END LOOP;
---
WHEN OTHERS THEN
po_error_msg := 'Others Exception raise in ' || func_name || ',' || SQLERRM;
END back_sad_cost_line_t;
END SAD_CALC_ITEM_PKG_TEST_OB;

Output
CREATE OR REPLACE PROCEDURE SAD.SAD_CALC_ITEM_PKG_TEST_OB#back_sad_cost_line_t ( pi_contract_number
IN VARCHAR2
,pi_quotation_id IN NUMBER
,pi_product_code IN VARCHAR2
,pi_process_batch_number IN NUMBER
,po_error_msg OUT VARCHAR2 ) IS MIG_PV_VAL_DUMMY_G_FUNC_NAME VARCHAR2 ( 30 ) :=
MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE ( current_schema ( )
,'SAD_CALC_ITEM_PKG_TEST_OB'
,'g_func_name' ) ::VARCHAR2 ( 30 ) ;
ex_data_error
EXCEPTION ;
ex_prog_error
EXCEPTION ;
BEGIN
---
LOOP
INSERT INTO sad_cost_line_bak ( 
processing_batch_number
,contract_number
,product_code
,quotation_id
,item_code
,refresh_date
,split_date
,SAD.SAD_CALC_ITEM_PKG_TEST_OB#error_msg ( )
,created_by
,creation_date
,last_updated_by
,last_update_date
 ) VALUES 
( pi_process_batch_number ,cur_1.contract_number ,cur_1.product_code ,cur_1.quotation_id ,cur_1.item_code
,cur_1.refresh_date ,cur_1.split_date ,cur_1.error_msg ,cur_1.created_by ,cur_1.creation_date ,cur_1.last_updated_by ,cur_1.last_update_date ) ;
END LOOP ;
---
WHEN OTHERS THEN
po_error_msg := 'Others Exception raise in ' || SAD.SAD_CALC_ITEM_PKG_TEST_OB#func_name ( ) || ',' || SQLERRM ;
END ;

Input
CREATE OR REPLACE PACKAGE BODY SAD.bas_dml_lookup_pkg IS
  g_pkg_name CONSTANT VARCHAR2(30) := 'bas_dml_ic_price_rule_pkg';
g_func_name VARCHAR2(100);

FUNCTION func_name
RETURN VARCHAR2
IS
  l_func_name VARCHAR2(100) ;
BEGIN
  l_func_name := g_pkg_name || '.' || g_func_name ;
  RETURN l_func_name ;
END ;

PROCEDURE data_change_logs ( pi_table_name IN VARCHAR2
, pi_table_key_columns IN VARCHAR2
, po_error_msg OUT VARCHAR2
)
IS
  BEGIN
  g_func_name := 'insert_fnd_data_change_logs_t';
  INSERT INTO fnd_data_change_logs_t
  ( logid,table_name,table_key_columns )
  VALUES
  ( fnd_data_change_logs_t_s.NEXTVAL
    , pi_table_name, pi_table_key_columns );
  EXCEPTION
  WHEN OTHERS THEN
  po_error_msg := 'Others Exception raise in ' || func_name || ',' || SQLERRM;
END data_change_logs;
END bas_dml_lookup_pkg;
/

Output

CREATE OR REPLACE FUNCTION SAD.bas_dml_lookup_pkg#func_name
RETURN VARCHAR2
IS
  MIG_PV_VAL_DUMMY_G_PKG_NAME VARCHAR2(30) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE
  ( 'SAD','BAS_DML_LOOKUP_PKG','G_PKG_NAME' )::VARCHAR2(30) ;
  MIG_PV_VAL_DUMMY_G_FUNC_NAME VARCHAR2(100) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE
  ( 'SAD','BAS_DML_LOOKUP_PKG','G_FUNC_NAME' )::VARCHAR2(100) ;
  l_func_name VARCHAR2(100) ;
BEGIN
  l_func_name := MIG_PV_VAL_DUMMY_G_PKG_NAME || '.' || MIG_PV_VAL_DUMMY_G_FUNC_NAME ;
  RETURN l_func_name ;
END ;
/
CREATE OR REPLACE PROCEDURE SAD.bas_dml_lookup_pkg#data_change_logs ( pi_table_name IN
 VARCHAR2
, pi_table_key_columns IN VARCHAR2
, po_error_msg OUT VARCHAR2
)
IS
  MIG_PV_VAL_DUMMY_G_FUNC_NAME VARCHAR2(30) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE
  ( 'SAD','BAS_DML_LOOKUP_PKG','G_FUNC_NAME' )::VARCHAR2(30) ;
BEGIN
  MIG_PV_VAL_DUMMY_G_FUNC_NAME := 'insert_fnd_data_change_logs_t' ;
  INSERT INTO fnd_data_change_logs_t
  ( logid,table_name,table_key_columns )
  VALUES ( NEXTVAL ( 'fnd_data_change_logs_t_s' )
    , pi_table_name,pi_table_key_columns ) ;
  MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE ( 'SAD','BAS_DML_LOOKUP_PKG','G_FUNC_NAME',
  MIG_PV_VAL_DUMMY_G_FUNC_NAME ) ;
  EXCEPTIION
  WHEN OTHERS THEN
  po_error_msg := 'Others Exception raise in ' || SAD.bas_dml_lookup_pkg#func_name( ) || ',' || SQLERRM;
END data_change_logs;
This section contains the migration syntax for migrating Oracle PL/SQL Collections. The migration syntax decides how the supported keywords/features are migrated.

A user-defined type (UDT) is a data type that is derived from a supported data type.

User-defined data types use built-in data types and other user-defined data types as the building blocks for data types that model the structure and behavior of data in applications. User-defined types make it easier to work with PL/SQL collections.

**UDT Table**

The table type is created to track the structure of the user-defined type. No data will be stored in the table.

**Input - CREATE TABLE TYPE**

```sql
CREATE <OR REPLACE> TYPE <schema.>inst_no_type IS TABLE OF VARCHAR2 (32767);
```

**Output**

```sql
CREATE TABLE<schema.>mig_inst_no_type
  ( typ_col VARCHAR2 (32767) );
```

**UDT VArray**

**Input - CREATE VArray**

```sql
CREATE TYPE phone_list_typ_demo AS VARRAY(n) OF VARCHAR2(25);
```

**Output**

```sql
CREATE TABLE mig_pone_list_typ_demo
  ( typ_col VARCHAR2 (25) );
```

**Declare user-defined type**

**Input - Declare user-defined type**

```sql
DECLARE
  v_sql_txt_array inst_no_type <:= inst_no_type();
BEGIN
  ...
END;
```

**Output**

```sql
DECLARE
  /*       v_sql_txt_array inst_no_type <:= inst_no_type()>; */
BEGIN
  EXECUTE IMMEDIATE 'DROP TABLE IF EXISTS
  v_sql_txt_array;
  CREATE LOCAL TEMPORARY TABLE
  v_sql_txt_array
```
ON COMMIT PRESERVE ROWS
AS SELECT *
    , CAST(NULL AS INT) AS
    typ_idx_col
FROM mig_inst_no_type
WHERE FALSE;

UDT Count

Input - UDT - COUNT in FOR LOOP
BEGIN
  ... 
  FOR i IN 1..v_jobnum_list.COUNT
  LOOP
    SELECT COUNT(*) INTO v_abc
    FROM ...
    WHERE ...
    AND nvl(t.batch_num, c_batchnum_null_num) = v_jobnum_list(i);
  ... 
  END LOOP;
  ... 

Output
BEGIN
  ... 
  FOR i IN 1..(SELECT COUNT(*) from v_jobnum_list)
  LOOP
    SELECT COUNT(*) INTO v_abc
    FROM ...
    WHERE ...
    AND nvl(t.batch_num, c_batchnum_null_num) = (SELECT typ_col FROM v_jobnum_list
    WHERE typ_idx_col = i);
  ... 
  END LOOP;
  ...

UDT Record

Record type is used to create records and can be defined in the declarative part of
any PL/SQL block, subprogram, or package.

Input - RECORD Type
Create
  or Replace Procedure test_proc AS TYPE t_log IS RECORD ( col1 int ,col2 emp.ename % type ) ;
  fr_wh_sql t_log ;
BEGIN
  fr_wh_sql.col1 := 101 ;
  fr_wh_sql.col2 := 'abcd' ;
  DBMS_OUTPUT.PUT_LINE ( fr_wh_sql.col1 || ',' || fr_wh_sql.col2 ) ;
END test_proc;
/

Output
Create
  or Replace Procedure test_proc AS /*TYPE t_log IS RECORD ( col1 int,col2 emp.ename%type );*/
  fr_wh_sql RECORD ;
  MIG_t_log col1 int ;
  MIG_t_log col2 emp.ename % type ;
BEGIN
  select
    MIG_t_log col1 as col1 ,MIG_t_log col2 as col2 INTO FR_WH_SQL ;
Enhancement of User-defined types

The tool supports the enhancement of PL/SQL type of TABLE used in Oracle for specific data types and for any table column.

**Input - PL/SQL type of TABLE of a specific data-type**

```sql
DECLARE
  type fr_wh_sql_info_type is table of VARCHAR(10);
  fr_wh_sql fr_wh_sql_info_type := fr_wh_sql_info_type();
BEGIN
  ...
END;
/
```

**Output**

```sql
DECLARE
  /* type fr_wh_sql_info_type is table of varchar(10); */
  /* fr_wh_sql fr_wh_sql_info_type := fr_wh_sql_info_type(); */
BEGIN
  EXECUTE IMMEDIATE 'DROP TABLE IF EXISTS mig_fr_wh_sql_info_type;
  CREATE LOCAL TEMPORARY TABLE mig_fr_wh_sql_info_type
  ( typ_col VARCHAR (10) )
  ON COMMIT PRESERVE ROWS';
  EXECUTE IMMEDIATE 'DROP TABLE IF EXISTS fr_wh_sql;
  CREATE LOCAL TEMPORARY TABLE fr_wh_sql
  ON COMMIT PRESERVE ROWS AS
  AS SELECT  *, CAST(NULL AS INT) AS typ_idx_col
  FROM mig_fr_wh_sql_info_type
  WHERE FALSE';
  ...
END;
/
```

**Input - PL/SQL type of TABLE of any table's column**

```sql
DECLARE
  type fr_wh_sql_info_type is table of fr_wh_sql_info.col1%type;
  fr_wh_sql fr_wh_sql_info_type := fr_wh_sql_info_type();
BEGIN
  ...
END;
/
```

**Output**

```sql
DECLARE
  /* type fr_wh_sql_info_type is table of fr_wh_sql_info.col1%type; */
  /* fr_wh_sql fr_wh_sql_info_type := fr_wh_sql_info_type(); */
BEGIN
  EXECUTE IMMEDIATE 'DROP TABLE IF EXISTS mig_fr_wh_sql_info_type;
  CREATE LOCAL TEMPORARY TABLE mig_fr_wh_sql_info_type
  ( typ_col VARCHAR (10) )
  ON COMMIT PRESERVE ROWS
  AS SELECT col1 AS typ_col
  FROM fr_wh_sql_info
  WHERE FALSE';
  ...
END;
/
```

**EXTEND**

Gauss SUPPORTS extend KEYWORD.
Input - Extend

FUNCTION FUNC_EXTEND ( in_str IN VARCHAR2 )
    RETURN ARRYTYPE AS
        v_count2 INTEGER;
        v_strlist arrytype;
        v_node VARCHAR2 (2000);
    BEGIN
        v_count2 := 0;
        v_strlist := arrytype ( ) ;
        FOR v_i IN 1 .. LENGTH (in_str)
            LOOP
                IF v_node IS NULL
                    THEN
                        v_node := '';
                        END IF;
                IF ( v_count2 = 0 ) OR ( v_count2 IS NULL )
                    THEN
                        EXIT ;
                        ELSE
                            v_strlist.EXTEND ( );
                            v_strlist ( v_i ) := v_node;
                            v_node := '';
                            END IF;
                    END LOOP ;
    RETURN v_strlist;
    END ;
/

Output

FUNCTION FUNC_EXTEND ( in_str IN VARCHAR2 )
    RETURN ARRYTYPE AS v_count2 INTEGER ;
    v_strlist arrytype ;
    v_node VARCHAR2 ( 2000 ) ;
    BEGIN
        v_count2 := 0 ;
        v_strlist := arrytype ( ) ;
        FOR v_i IN 1 .. LENGTH( in_str ) LOOP
            IF v_node IS NULL
                THEN
                    v_node := '';
                    END IF ;
            IF ( v_count2 = 0 )
                OR( v_count2 IS NULL )
                THEN
                    EXIT ;
                ELSE
                    v_strlist.EXTEND ( 1 ) ;
                    v_strlist ( v_i ) := v_node ;
                    v_node := '';
                    END IF ;
        END LOOP ;
    RETURN v_strlist;
    END ;
/

RECORD

Record type declared in Package specification is actually declared as type in the corresponding Package body.

It depends on the following configuration item:
**plsqlCollection=varray**

**Input – RECORD**

```
CREATE OR REPLACE FUNCTION func1 (i1 INT)
RETURN INT
AS
    TYPE r_rthpagat_list IS RECORD (--Record information about cross-border RMB business parameters (rthpagat)
        rthpagat_REQUESTID RMTS_REMITTANCE_PARAM.REQUESTID%TYPE ,rthpagat_PARAMTNAME
        RMTS_REMITTANCE_PARAM.PARAMTNAME%TYPE ,rthpagat_PARAMNUM
        RMTS_REMITTANCE_PARAM.PARAMNUM%TYPE ,rthpagat_PARAMSTAT
        RMTS_REMITTANCE_PARAM.PARAMSTAT%TYPE ,rthpagat_REQTELLERNO
        RMTS_REMITTANCE_PARAM.REQTELLERNO%TYPE ,rthpagat_REQUESTTIME
        RMTS_REMITTANCE_PARAM.REQUESTTIME%TYPE ,rthpagat_HOSTERRNO
        RMTS_REMITTANCE_PARAM.HOSTERRNO%TYPE ,rthpagat_HOSTERRMSG
        RMTS_REMITTANCE_PARAM.HOSTERRMSG%TYPE ,rthpagat_GATBANK
        RMTS_REMITTANCE_PARAM.GATBANK%TYPE ,rthpagat_GATEEBANK
        RMTS_REMITTANCE_PARAM.VALUE1%TYPE ,rthpagat_TEL
        RMTS_REMITTANCE_PARAM.VALUE3%TYPE ,rthpagat_DATE
        RMTS_REMITTANCE_PARAM.VALUE4%TYPE ,rthpagat_BM_GATBANK
        RMTS_REMITTANCE_PARAM.VALUE5%TYPE ,rthpagat_BM_GATEEBANK
        RMTS_REMITTANCE_PARAM.VALUE6%TYPE ,rthpagat_BM_LMTEL
        RMTS_REMITTANCE_PARAM.VALUE7%TYPE ,rthpagat_BM_LMDAT
        RMTS_REMITTANCE_PARAM.VALUE8%TYPE ) ;

    v1  r_rthpagat_list;
BEGIN
END;
```

**Output**

```
CREATE
TYPE rmts_remitparammgmt_rthpagat.r_rthpagat_list AS /* O_ERRMSG error description */
Rthpagat_REQUESTID
    rthpagat_REQUESTID RMTS_REMITTANCE_PARAM.REQUESTID%TYPE ,rthpagat_PARAMTNAME
    RMTS_REMITTANCE_PARAM.PARAMTNAME%TYPE ,rthpagat_PARAMNUM
    RMTS_REMITTANCE_PARAM.PARAMNUM%TYPE ,rthpagat_PARAMSTAT
    RMTS_REMITTANCE_PARAM.PARAMSTAT%TYPE ,rthpagat_REQTELLERNO
    RMTS_REMITTANCE_PARAM.REQTELLERNO%TYPE ,rthpagat_REQUESTTIME
    RMTS_REMITTANCE_PARAM.REQUESTTIME%TYPE ,rthpagat_HOSTERRNO
    RMTS_REMITTANCE_PARAM.HOSTERRNO%TYPE ,rthpagat_HOSTERRMSG
    RMTS_REMITTANCE_PARAM.HOSTERRMSG%TYPE ,rthpagat_GATBANK
    RMTS_REMITTANCE_PARAM.GATBANK%TYPE ,rthpagat_GATEEBANK
    RMTS_REMITTANCE_PARAM.VALUE1%TYPE ,rthpagat_TEL
    RMTS_REMITTANCE_PARAM.VALUE3%TYPE ,rthpagat_DATE
    RMTS_REMITTANCE_PARAM.VALUE4%TYPE ,rthpagat_BM_GATBANK
    RMTS_REMITTANCE_PARAM.VALUE5%TYPE ,rthpagat_BM_GATEEBANK
    RMTS_REMITTANCE_PARAM.VALUE6%TYPE ,rthpagat_BM_LMTEL
    RMTS_REMITTANCE_PARAM.VALUE7%TYPE ,rthpagat_BM_LMDAT
    RMTS_REMITTANCE_PARAM.VALUE8%TYPE ) ;

    v1  r_rthpagat_list;
BEGIN
END;
```

**Naming convention of TYPE Name**

User-defined types allow for the definition of data types that model the structure and behavior of the data in an application.
CREATE TYPE t_line AS ( product_line VARCHAR2 ( 30 ), product_amount NUMBER );

Output
CREATE TYPE sad_dml_product_pkg.t_line AS ( product_line VARCHAR2 ( 30 ), product_amount NUMBER ) ;

Input
CREATE TYPE t_line AS ( product_line VARCHAR2 ( 30 ), product_amount NUMBER ) ;

Output
CREATE TYPE SAD.sad_dml_product_pkg#t_line AS ( product_line VARCHAR2 ( 30 ), product_amount NUMBER ) ;

NOTE
- For the first output (pkg.t), if "pkgSchemaNaming" is set to "true" in config file, PL RECORD migration should have package name as schema name along with type name.
- For the second output (pkg#t) Assume TYPE belongs to sad_dml_product_pkg package. If "pkgSchemaNaming" is set to false in config file, PL RECORD migration should have schema name as schema name along with package name + type name separated by # as type name.

SUBTYPE

With the SUBTYPE statement, PL/SQL allows you to define your own subtypes or aliases of predefined data types, sometimes referred to as abstract data types.

Input
CREATE OR REPLACE PACKAGE "SAD"."BAS_SUBTYPE_PKG" IS
SUBTYPE CURRENCY IS BAS_PRICE_LIST_T.CURRENCY%TYPE;
END bas_subtype_pkg;
/
CREATE OR REPLACE PACKAGE BODY "SAD"."BAS_SUBTYPE_PKG" IS
BEGIN
NULL;
END bas_subtype_pkg;
/
--********************************************************************
CREATE OR REPLACE PACKAGE BODY SAD.bas_lookup_misc_pkg IS
FUNCTION get_currency(pi_price_type IN NUMBER) RETURN VARCHAR2 IS
v_currency bas_subtype_pkg.currency;
BEGIN
  g_func_name := 'get_currency';
  FOR rec_currency IN (SELECT currency FROM sad_price_type_v WHERE price_type_code = pi_price_type)
  LOOP
    v_currency := rec_currency.currency;
    END LOOP;
RETURN v_currency;
END get_currency;
END SAD.bas_lookup_misc_pkg;
/

Output

Data Warehouse Service
Tool Guide
6 Migration Tool - SQL Syntax Migration Tool

Issue 03 (2019-08-08) Copyright © Huawei Technologies Co., Ltd. 549
CREATE OR REPLACE FUNCTION SAD.bas_lookup_misc_pk#get_currency(pi_price_type IN NUMBER) RETURN VARCHAR2 IS
  v_currency BAS_PRICE_LIST_T.CURRENCY%TYPE;
BEGIN
  g_func_name := 'get_currency';
  FOR rec_currency IN (SELECT currency FROM sad_price_type_v WHERE price_type_code = pi_price_type)
  LOOP
    v_currency := rec_currency.currency;
  END LOOP;
  RETURN v_currency;
END ; /

As SUBTYPE not supported in Gauss, where the SUBTYPE variable is used need to replace with the actual type which is used on SUBTYPE creation

### 6.13.11 PL/SQL Packages

#### 6.13.11.1 Overview

This section contains the migration syntax for migrating Oracle PL/SQL Packages and REF CURSOR. The migration syntax decides how the supported keywords/features are migrated.

#### 6.13.11.2 Packages

Packages are schema objects that group logically related PL/SQL types, variables, functions and procedures. In Oracle, each package consists of two parts: package specification and package body. The package specification may contain variables, including REF CURSOR, declared in it. The package REF CURSORS are identified and migrated in the referred places. The functions and procedures in the package body are migrated into individual functions and procedures. The types and variables in the package body are migrated to within each of the functions and procedures.

If the schema name of the package specification and package body does not match, then the tool will log a schema name mismatch error to the migrationtoolError.log file.
**Figure 6-59 Migration of PL/SQL Packages**

**Input - PL/SQL Packages** (Package Specification and Body)

```
CREATE or REPLACE PACKAGE BODY pkg_get_empdet
IS
    PROCEDURE get_ename(eno in number, ename out varchar2) IS
        BEGIN
            SELECT ename || ',' || last_name
            INTO ename
            FROM emp
            WHERE empno = eno;
        END get_ename;

    FUNCTION get_sal(eno in number) return number IS
        lsalary number;
        BEGIN
            SELECT salary
            INTO lsalary
            FROM emp
            WHERE empno = eno;
            RETURN lsalary;
        END get_sal;
    END pkg_get_empdet;
/
```

**Output** (The output contains separate functions and procedures for each of the functions and procedures in the package body of the input)

```
CREATE 
or REPLACE PROCEDURE 
pkg_get_empdet.get_ename ( eno in number , ename out varchar2 ) IS
    BEGIN
        SELECT ename || ',' || last_name INTO ename
        FROM emp
        WHERE empno = eno;
    END get_ename;
```
CREATE or REPLACE FUNCTION pkg_get_empdet.get_sal ( eno in number )
return number IS
    lsalary number ;
BEGIN
    SELECT
        salary INTO lsalary
    FROM
        emp
    WHERE
        empno = eno ;
    RETURN lsalary ;
END ;
/

Input - PL/SQL Packages
CREATE OR replace VIEW vw_emp_name AS (SELECT
    pkg_get_empdet.get_sal (emp.empno) AS empsal
FROM
    emp)
;

Output
CREATE OR replace VIEW vw_emp_name AS (SELECT
    pkg_get_empdet.get_sal (emp.empno) AS empsal
FROM
    emp)
;

set
package_name_list = 'func' ;
CREATE OR REPLACE FUNCTION func1 ( i1 INT )
RETURN INT As TYPE r_rthpagat_list IS RECORD ( /* Record
    information about cross-border RMB */
    business parameters ( rthpagat ) rthpagat REQUESTID
    RMTS_REMITTANCE_PARAM.REQUESTID%TYPE ,rthpagat PARAMTNAME
    RMTS_REMITTANCE_PARAM.PARAMTNAME%TYPE ,rthpagat PARAMNUM
    RMTS_REMITTANCE_PARAM.PARAMNUM%TYPE ,rthpagat PARAMSTAT
    RMTS_REMITTANCE_PARAM.PARAMSTAT%TYPE ,rthpagat REQTELLERNO
    RMTS_REMITTANCE_PARAM.REQTELLERNO%TYPE ) ,rthpagat REQUESTTIME
    RMTS_REMITTANCE_PARAMREQUESTTIME%TYPE
    rthpagat HOSTERRNO RMTS_REMITTANCE PARAMHOSTERRNO%TYPE ,rthpagat HOSTERRMSG
    RMTS_REMITTANCE_PARAM.HOSTERRMSG%TYPE ,rthpagat GATBANK
    RMTS_REMITTANCE PARAM.VALUE1%TYPE ,rthpagat GATEEBANK
    RMTS_REMITTANCE_PARAM.VALUE2%TYPE ,rthpagat TELLER
    RMTS_REMITTANCE PARAM.VALUE3%TYPE ,rthpagat DATE
    RMTS_REMITTANCE PARAM.VALUE4%TYPE ,rthpagat BM_GATBANK
    RMTS_REMITTANCE_PARAM.VALUE5%TYPE ,rthpagat BM_GATEEBANK
    RMTS_REMITTANCE_PARAM.VALUE6%TYPE ,rthpagat BM_LMTEL
    RMTS_REMITTANCE_PARAM.VALUE7%TYPE ,rthpagat BM_LMDAT
    RMTS_REMITTANCE_PARAM.VALUE8%TYPE ) ;
    v1 r_rthpagat_list ;
BEGIN
    END ;
    /
    reset
package_name_list ;
Input - Function/Procedure With No Parameter

In case, a procedure or function does not have any parameter or argument, need to put () after procedure or function name while calling the same.

```sql
CREATE OR REPLACE PACKAGE BODY SAD.bas_lookup_misc_pkg IS
  g_pkg_name CONSTANT VARCHAR2(30) := 'bas_lookup_misc_pkg';
  g_func_name VARCHAR2(30);

  FUNCTION func_name
  RETURN VARCHAR2
  IS
    l_func_name VARCHAR2(100);
  BEGIN
    l_func_name := g_pkg_name || '.' || g_func_name;
    RETURN l_func_name;
  END func_name;

  PROCEDURE insert_fnd_data_change_logs(pi_table_name               IN VARCHAR2, 
                                         pi_table_key_columns        IN VARCHAR2, 
                                         pi_table_key_values         IN VARCHAR2, 
                                         pi_column_name              IN VARCHAR2, 
                                         pi_column_change_from_value IN VARCHAR2, 
                                         pi_column_change_to_value   IN VARCHAR2, 
                                         pi_op_code                  IN NUMBER, 
                                         pi_description              IN VARCHAR2, 
                                         po_error_msg                OUT VARCHAR2)
  IS
    g_func_name := 'insert_fnd_data_change_logs_t';
  BEGIN
    l_func_name := g_pkg_name || '.' || g_func_name;
    RETURN l_func_name;
  END;

END SAD.bas_lookup_misc_pkg;
/
```

Output

```sql
CREATE OR REPLACE FUNCTION SAD.bas_lookup_misc_pkg#func_name
  RETURN VARCHAR2
PACKAGE
IS
  l_func_name VARCHAR2 (100);
  MIG_PV_VAL_DUMMY_G_PKG_NAME VARCHAR2 (30) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE ('SAD', 'bas_lookup_misc_pkg', 'g_pkg_name') :: VARCHAR2 (30);
  MIG_PV_VAL_DUMMY_G_FUNC_NAME VARCHAR2 (30) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE ('SAD', 'bas_lookup_misc_pkg', 'g_func_name') :: VARCHAR2 (30);
  BEGIN
    l_func_name := MIG_PV_VAL_DUMMY_G_PKG_NAME || '.' || MIG_PV_VAL_DUMMY_G_FUNC_NAME;
    RETURN l_func_name;
  END;
```

---

Data Warehouse Service
Tool Guide
6 Migration Tool - SQL Syntax Migration Tool

Issue 03 (2019-08-08)  Copyright © Huawei Technologies Co., Ltd.
OR REPLACE PROCEDURE SAD.bas_lookup_misc_pkg#insert_fnd_data_change_logs ( pi_table_name IN VARCHAR2,
pi_table_key_columns IN VARCHAR2,
pi_table_key_values IN VARCHAR2,
pi_column_name IN VARCHAR2,
pi_column_change_from_value IN VARCHAR2,
pi_column_change_to_value IN VARCHAR2,
pi_op_code IN NUMBER,
pi_description IN VARCHAR2,
po_error_msg OUT VARCHAR2 )

PACKAGE
IS
  MIG_PV_VAL_DUMMY_G_FUNC_NAME VARCHAR2 ( 30 ) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE( 'SAD' , 'bas_lookup_misc_pkg' , 'g_func_name' ) :: VARCHAR2 ( 30 ) ;

BEGIN
  MIG_PV_VAL_DUMMY_G_FUNC_NAME := 'insert_fnd_data_change_logs_t' ;
  MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE( 'SAD' , 'bas_lookup_misc_pkg' , 'g_pkg_name' , MIG_PV_VAL_DUMMY_G_PKG_NAME ) ;
  MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE( 'SAD' , 'bas_lookup_misc_pkg' , 'g_func_name' , MIG_PV_VAL_DUMMY_G_FUNC_NAME ) ;

EXCEPTION
  WHEN OTHERS THEN
    po_error_msg := 'Others Exception raise in ' || SAD.bas_lookup_misc_pkg#func_name() || ',' || SQLERRM ;

END ;
/

Input - Package Body with no procedure and functions

In case package body does not have any logic, that is procedures and functions, then, migration tool needs to remove all code from the same package. Basically output should be blank.

CREATE OR REPLACE PACKAGE BODY SAD.bas_subtype_pkg IS
BEGIN
  NULL;
END bas_subtype_pkg;
/

Input - SUBTYPE

With the SUBTYPE statement, PL/SQL allows you to define your own subtypes or aliases of predefined data types, sometimes referred to as abstract data types.

CREATE OR REPLACE PACKAGE "SAD"."BAS_SUBTYPE_PKG" IS
  SUBTYPE CURRENCY IS BAS_PRICE_LIST_T.CURRENCY%TYPE;
END bas_subtype_pkg;
/
CREATE OR REPLACE PACKAGE BODY "SAD"."BAS_SUBTYPE_PKG" IS
BEGIN
  NULL;
END bas_subtype_pkg;
/
***-----------------------------------------------------------------------
CREATE OR REPLACE PACKAGE BODY SAD.bas_lookup_misc_pkg IS
  FUNCTION get_currency(pi_price_type IN NUMBER) RETURN VARCHAR2 IS
    v_currency bas_subtype_pkg.currency;
    BEGIN
      g_func_name := 'get_currency';
      FOR rec_currency IN (SELECT currency FROM sad_price_type_v WHERE price_type_code = pi_price_type)
      LOOP
        v_currency := rec_currency.currency;
      END LOOP;
      RETURN v_currency;
    END;
END bas_subtype_pkg;
RETURN v_currency;
END get_currency;
END SAD.bas_lookup_misc_pkg;
/

Output

"SAD":"BAS_SUBTYPE_PKG" package will be blank after migration.
---
CREATE OR REPLACE FUNCTION SAD.bas_lookup_misc_pkg#get_currency(pi_price_type IN NUMBER)
RETURN VARCHAR2 IS
  v_currency BAS_Price_LIST_T.CURRENCY%TYPE;
BEGIN
  g_func_name := 'get_currency';
  FOR rec_currency IN (SELECT currency FROM sad_price_type_v WHERE price_type_code = pi_price_type)
  LOOP
    v_currency := rec_currency.currency;
    END LOOP;
  RETURN v_currency;
END;
/

NOTE

As SUBTYPE not supported in Gauss, where the SUBTYPE variable is used need to replace with the actually type which is used on SUBTYPE creation.

Input - sys.dbms_job

The DBMS_JOB package schedules and manages jobs in the job queue.

CREATE OR replace PACKAGE BODY "SAD"."EIP_HTM_INTEGRATION_PKG"
IS
  PROCEDURE Greate_import_instruction_job
  IS
    v_jobid NUMBER;
    BEGIN
      IF bas_lookup_misc_pkg#Exits_run_job('eip_htm_integration_pkg.import_instruction_job') = 'N' THEN
        sys.dbms_job.Submit(job => v_jobid,
          what => 'begin
            eip_htm_integration_pkg.import_instruction_job;
          end';
          next_date => SYSDATE);
        COMMIT;
      END IF;
      ---
      END greate_import_instruction_job;
      END eip_htm_integration_pkg;
      /

Output

CREATE OR replace PROCEDURE
sad.Eip_htm_integration_pkg#greate_import_instruction_job
IS
  v_jobid NUMBER;
  BEGIN
    IF Bas_lookup_misc_pkg#Exits_run_job('eip_htm_integration_pkg.import_instruction_job') = 'N' THEN
      dbms_job.Submit(job => v_jobid,
        what => 'begin
          eip_htm_integration_pkg.import_instruction_job;
        end';
        next_date => SYSDATE);
      /* COMMIT; */
      END IF;
      /* COMMIT; */
      NULL;
      END IF;
      /* COMMIT; */
Need to remove SYS schema while calling the package.

**Input - Procedure/Function variable**

As NULL constraint not supported on variable declaration by Gauss, need to comment the NULL keyword.

```sql
CREATE OR REPLACE PACKAGE BODY SAD.sad_lookup_contract_pkg IS
  FUNCTION CONTRACT_DISTRIBUTE_STATUS_S2(PI_CONTRACT_NUMBER IN VARCHAR2)
  RETURN VARCHAR2 IS
    L_CONTRACT_DISTRIBUTE_STATUS VARCHAR2(10)  NULL;
  BEGIN
    IF CUR_CONTRACT.CONTRACT_STATUS = 0 THEN
      L_CONTRACT_DISTRIBUTE_STATUS := 'Cancel';
    ELSE
      L_CONTRACT_DISTRIBUTE_STATUS := 'Active';
    END IF;
    RETURN L_CONTRACT_DISTRIBUTE_STATUS;
  EXCEPTION
    WHEN OTHERS THEN
      L_CONTRACT_DISTRIBUTE_STATUS := NULL;
  END CONTRACT_DISTRIBUTE_STATUS_S2;
END sad_lookup_contract_pkg;
/
```

**Output**

```sql
CREATE OR replace FUNCTION sad_lookup_contract_pkg.Contract_distribute_status_s2
( pi_contract_number IN VARCHAR2 )
RETURN VARCHAR2
IS
  l_contract_distribute_statusvarchar2 ( 10 ) /* NULL */;
BEGIN
  IF cur_contract.contract_status = 0 THEN
    l_contract_distribute_status := 'Cancel' ;
  ELSE
    l_contract_distribute_status := 'Active' ;
  END IF ;
  RETURN l_contract_distribute_status ;
EXCEPTION
  WHEN OTHERS THEN
    l_contract_distribute_status := NULL ;
END ;/
```

**Input - Configuration parameter addPackageNameList = true**

Hint to access objects from specific schema by system.

```sql
CREATE OR REPLACE PACKAGE BODY IC_STAGE.PKG_REVN_ARPU
AS
---------------------
END PKG_REVN_ARPU;
/
```

**Output**

```sql
SET  package_name_list = 'PKG_REVN_ARPU' ;
---------------------
```
Input - Configuration parameter `addPackageNameList = false`  
Hint to access objects from specific schema by system.

```
CREATE OR REPLACE PACKAGE BODY IC_STAGE.PKG_REVN_ARPU AS
 ------------
END PKG_REVN_ARPU;
/
```

Output  
```
SET SEARCH_PATH=PKG_REVN_ARPU,PUBLIC;
```

Input - PACKAGE  
Hints for procedure and functions belong to package.

```
CREATE OR REPLACE PACKAGE BODY SAD.sad_lookup_contract_pkg IS
 FUNCTION CONTRACT_DISTRIBUTE_STATUS_S2(PI_CONTRACT_NUMBER IN VARCHAR2)
 RETURN VARCHAR2 IS
 L_CONTRACT_DISTRIBUTE_STATUS VARCHAR2(10) ;
 BEGIN
  IF CUR_CONTRACT.CONTRACT_STATUS = 0 THEN
   L_CONTRACT_DISTRIBUTE_STATUS := 'Cancel';
  ELSE
   L_CONTRACT_DISTRIBUTE_STATUS := 'Active';
  END IF;
  RETURN L_CONTRACT_DISTRIBUTE_STATUS;
EXCEPTION
 WHEN OTHERS THEN
  L_CONTRACT_DISTRIBUTE_STATUS := NULL;
 END CONTRACT_DISTRIBUTE_STATUS_S2;
END sad_lookup_contract_pkg;
/
```

Output  
```
CREATE OR replace FUNCTION sad_lookup_contract_pkg.Contract_distribute_status_s2
( pi_contract_number IN VARCHAR2 )
RETURN VARCHAR2
PACKAGE
IS
  l_contract_distribute_statusvarchar2 ( 10 ) ;
BEGIN
  IF cur_contract.contract_status = 0 THEN
    l_contract_distribute_status := 'Cancel';
  ELSE
    l_contract_distribute_status := 'Active';
  END IF ;
  RETURN l_contract_distribute_status ;
EXCEPTION
 WHEN OTHERS THEN
  l_contract_distribute_status := NULL ;
 END ;
/
```

**NOTE**  
Need to put PACKAGE keyword while creating any procedure and function before IS/AS statement.
### Input - Nested Procedure

Creating procedure inside a procedure is known as Nested procedure. Nested procedures are private, scope is within parent procedure.

```sql
CREATE OR REPLACE PROCEDURE refresh_sw_product_amount(pi_stage_id IN NUMBER) IS
    v_product_amount sad_sw_product_amount_t.product_amount%TYPE;
FUNCTION get_sw_no RETURN VARCHAR2 IS
    v_xh NUMBER;
BEGIN
    BEGIN
        SELECT nvl(to_number(substrb(MAX(sw_no), 3, 4)), 0)
        INTO v_xh
        FROM sad.sad_sw_product_amount_t
        WHERE pi_stage_id = pi_stage_id;
    EXCEPTION WHEN OTHERS THEN
        v_xh := 0;
    END;
    RETURN 'SW' || lpad(to_char(v_xh + 1), 4, '0') || 'Y';
END get_sw_no;
BEGIN
    FOR rec_pu IN (SELECT t.*, sh.header_id
        FROM asms.ht_stages t, asms.ht, sad.sad_distribution_headers_t sh
        WHERE t.hth = ht.hth
        AND sh.contract_number = t.hth
        AND sh.stage_id = t.stage_id
        AND t.category_id IN
            (SELECT code
                FROM asms.asms_lookup_values
                WHERE type_code = 'CATEGORY_ID_EQUIPMENT'
                AND enabled_flag = 'Y')
        AND nvl(t.status, '-1') <> '0'
        AND t.stage_id = pi_stage_id)
    LOOP
        SELECT nvl(SUM(nvl(product_amount, 0)), 0)
        INTO v_product_amount
        FROM sad.sad_products_t sp
        WHERE sp.header_id = rec_pu.header_id
        AND sp.sw_flag = 'Y';
    END LOOP;
END refresh_sw_product_amount;
```

### Output

```sql
CREATE OR REPLACE FUNCTION get_sw_no(pi_stage_id IN NUMBER) RETURN VARCHAR2 IS
    v_xh NUMBER;
BEGIN
    BEGIN
        SELECT nvl(to_number(substrb(MAX(sw_no), 3, 4)), 0)
        INTO v_xh
        FROM sad.sad_sw_product_amount_t
        WHERE pi_stage_id = pi_stage_id;
    EXCEPTION WHEN OTHERS THEN
        v_xh := 0;
    END;
```
CREATE OR REPLACE PROCEDURE refresh_sw_product_amount(pi_stage_id IN NUMBER) IS
  v_product_amount          sad_sw_product_amount_t.product_amount%TYPE;
BEGIN
  FOR rec_pu IN (SELECT t.*, sh.header_id
                  FROM asms.ht_stages t, asms.ht, sad.sad_distribution_headers_t sh
                  WHERE t.hth = ht.hth
                  AND sh.contract_number = t.hth
                  AND sh.stage_id = t.stage_id
                  AND ht.sw_track_flag = 'Y'
                  AND to_char(t.category_id) IN
                    (SELECT code
                     FROM asms.asms_lookup_values
                     WHERE type_code = 'CATEGORY_ID_EQUIPMENT'
                     AND enabled_flag = 'Y')
                  AND nvl(t.status, '-1') <> '0'
                  AND t.stage_id = pi_stage_id)
  LOOP
    SELECT nvl(SUM(nvl(product_amount, 0)), 0)
    INTO v_product_amount
    FROM sad.sad_products_t sp
    WHERE sp.header_id = rec_pu.header_id
    AND sp.sw_flag = 'Y';
  END LOOP;
END;
/

When nested proc/functions are implemented, the package variables in all proc/functions must be processed.

The sub-procedures/function need to migrate followed by parent procedure/function

if "pkgSchemaNaming" = false

if "pkgSchemaNaming" is set to false, PL RECORD migration should not have package name in the type name as its schema.

Input

CREATE OR REPLACE PACKAGE BODY SAD.sad_dml_product_pkg IS
PROCEDURE save_sad_product_line_amount(pi_stage_id            IN NUMBER,
                                        pi_product_line_code   IN VARCHAR2,
                                        po_error_msg           OUT VARCHAR2) IS
  TYPE t_line IS RECORD(
    product_line   VARCHAR2(30),
    product_amount NUMBER);
  TYPE tab_line IS TABLE OF t_line INDEX BY BINARY_INTEGER;
  rec_line             tab_line;
  v_product_line_arr   VARCHAR2(5000);
  v_product_line       VARCHAR2(30) ;
  v_count              INTEGER;
  v_start              INTEGER;
  v_pos                INTEGER;
BEGIN

DECLARE
  v_count      := 0;
  v_start      := 1;
  v_product_line_arr := pi_product_line_code;
  v_pos := instr(v_product_line_arr, ',', v_start);
  IF v_pos <= 0
    THEN
      EXIT;
    END IF;
  v_product_line := substr(v_product_line_arr, v_start, v_pos - 1);
  v_count := v_count + 1;
  rec_line(v_count).product_line := v_product_line;
  rec_line(v_count).product_amount := 0;
  v_product_line_arr := substr(v_product_line_arr, v_pos + 1, length(v_product_line_arr));
END LOOP;

FOR v_count IN 1 .. rec_line.count
  LOOP
    UPDATE sad_product_line_amount_t spl
      SET spl.product_line_amount = rec_line(v_count).product_amount
      WHERE spl.stage_id = pi_stage_id
        AND spl.product_line_code = rec_line(v_count).product_line;
    IF SQL%NOTFOUND
      THEN
        INSERT INTO sad_product_line_amount_t
              (stage_id, product_line_code, product_line_amount)
            VALUES (pi_stage_id, rec_line(v_count).product_line, rec_line(v_count).product_amount);
      END IF;
    END LOOP;
  END LOOP;
END sad_dml_product_pkg;
/

CREATE  TYPE SAD.sad_dml_product_pkg#t_line AS
  ( product_line VARCHAR2 ( 30 )
    , product_amount NUMBER ) ;

CREATE OR REPLACE PROCEDURE SAD.sad_dml_product_pkg#save_sad_product_line_amount
  ( pi_stage_id IN NUMBER
    , pi_product_line_code IN VARCHAR2
    , po_error_msg OUT VARCHAR2 )
PACKAGE
IS
  TYPE tab_line IS VARRAY ( 10240 ) OF SAD.sad_dml_product_pkg#t_line;
  rec_line tab_line ;
  v_product_line_arr VARCHAR2 ( 5000 ) ;
  v_product_line VARCHAR2 ( 30 ) ;
  v_count INTEGER ;
  v_start INTEGER ;
  v_pos INTEGER ;
BEGIN
  v_count := 0 ;
  v_start := 1 ;
  v_product_line_arr := pi_product_line_code ;
  LOOP
    v_pos := instr( v_product_line_arr , ',' , v_start );
    IF v_pos <= 0 THEN
      EXIT ;
    END IF ;
  END LOOP ;

Output

CREATE  TYPE SAD.sad_dml_product_pkg#t_line AS
  ( product_line VARCHAR2 ( 30 )
    , product_amount NUMBER ) ;

CREATE OR REPLACE PROCEDURE SAD.sad_dml_product_pkg#save_sad_product_line_amount
  ( pi_stage_id IN NUMBER
    , pi_product_line_code IN VARCHAR2
    , po_error_msg OUT VARCHAR2 )
PACKAGE
IS
  TYPE tab_line IS VARRAY ( 10240 ) OF SAD.sad_dml_product_pkg#t_line;
  rec_line tab_line ;
  v_product_line_arr VARCHAR2 ( 5000 ) ;
  v_product_line VARCHAR2 ( 30 ) ;
  v_count INTEGER ;
  v_start INTEGER ;
  v_pos INTEGER ;
BEGIN
  v_count := 0 ;
  v_start := 1 ;
  v_product_line_arr := pi_product_line_code ;
  LOOP
    v_pos := instr( v_product_line_arr , ',' , v_start );
    IF v_pos <= 0 THEN
      EXIT ;
    END IF ;
v_product_line := SUBSTR ( v_product_line_arr , v_start , v_pos - 1 ) ;
v_count := v_count + 1 ;
rec_line ( v_count ).product_line := v_product_line ;
rec_line ( v_count ).product_amount := 0 ;
v_product_line_arr := SUBSTR ( v_product_line_arr , v_pos + 1 , length ( v_product_line_arr ) ) ;
END LOOP ;

FOR v_count IN 1 .. rec_line.count LOOP
  UPDATE sad_product_line_amount_t spl
  SET spl.product_line_amount = rec_line ( v_count ).product_amount
  WHERE spl.stage_id = pi_stage_id
  AND spl.product_line_code = rec_line ( v_count ).product_line ;
  IF SQL%NOTFOUND THEN
    INSERT INTO sad_product_line_amount_t
    ( stage_id, product_line_code, product_line_amount )
    VALUES ( pi_stage_id, rec_line ( v_count ).product_line
     , rec_line ( v_count ).product_amount ) ;
  END IF ;
END LOOP ;

EXCEPTION
  WHEN OTHERS THEN
    po_error_msg := 'Others Exception raise in ' || func_name || ',' || SQLERRM ;
END ;
/

6.13.11.3 Package Variables

Package variables are available in Oracle packages that allow variables to retain all the functions and procedures in the package. Migration Tool supports package variables for DWS using custom function.

**Note**

- **Prerequisites**
  - Create and use the MIG_ORA_EXT schema.
  - Copy the contents of the custom script and execute the script in all target databases for which migration is to be performed. For details, see Custom DB Script Configuration and Migration Process.
  - If there is a space between a schema name and a package name, or either the package specification or body has quotes, the output may not be as expected.

**Input - Package variables**

```sql
CREATE OR REPLACE PACKAGE scott.pkg_adm_util IS
  un_stand_value long := 'A';
defaultdate date := sysdate ;
g_pkgname CONSTANT VARCHAR2 ( 255 ) DEFAULT 'pkg_adm_util';
procedure p1 ;
END pkg_adm_util ;
/

CREATE OR REPLACE PACKAGE BODY scott.pkg_adm_util AS
defaulttime timestamp := systimestamp ;
PROCEDURE P1 AS BEGIN
  scott.pkg_adm_util.un_stand_value := 'A';
  pkg_adm_util.un_stand_value := 'B';
  un_stand_value := 'C';
  DBMS_OUTPUT.PUT_LINE ( pkg_adm_util.defaultdate ) ;
END ;
```
DBMS_OUTPUT.PUT_LINE ( defaulttime );
DBMS_OUTPUT.PUT_LINE ( scott.pkg_adm_util.un_stand_value );
DBMS_OUTPUT.PUT_LINE ( pkg_adm_util.un_stand_value );
DBMS_OUTPUT.PUT_LINE ( un_stand_value );
END;
END;
/

Output

SCHEMA pkg_adm_util
;
BEGIN
INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES
( SCHEMA_NAME,PACKAGE_NAME,SPEC_OR_BODY,VARIABLE_NAME,VARIABLE_TYPE,CONSTANT_I,DEFAULT_VALUE,EXPRESSION_I )
VALUES
( UPPER( 'scott' ),UPPER( 'pkg_adm_util' ),'S',UPPER('un_stand_value' ),UPPER( 'TEXT' ),false ,'''',false );
INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES
( SCHEMA_NAME,PACKAGE_NAME,SPEC_OR_BODY,VARIABLE_NAME,VARIABLE_TYPE,CONSTANT_I,DEFAULT_VALUE,EXPRESSION_I )
VALUES
( UPPER( 'scott' ),UPPER( 'pkg_adm_util' ),'S',UPPER('defaultdate' ),UPPER( 'DATE' ),false ,$q$sysdate$q$ ,false );
INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES
( SCHEMA_NAME,PACKAGE_NAME,SPEC_OR_BODY,VARIABLE_NAME,VARIABLE_TYPE,CONSTANT_I,DEFAULT_VALUE,EXPRESSION_I )
VALUES
( UPPER( 'scott' ),UPPER( 'pkg_adm_util' ),'S',UPPER('g_pkgname' ),UPPER( 'VARCHAR2' ) ,false ,'pkg_adm_util' ,true );
END;
BEGIN
INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES
( SCHEMA_NAME,PACKAGE_NAME,SPEC_OR_BODY,VARIABLE_NAME,VARIABLE_TYPE,CONSTANT_I,DEFAULT_VALUE,EXPRESSION_I )
VALUES
( UPPER( 'scott' ),UPPER( 'pkg_adm_util' ),'B',UPPER('defaulttime' ),UPPER( 'TIMESTAMP' ),false ,$q$CURRENT_TIMESTAMP$q$ ,true );
END;
/
CREATE OR REPLACE PROCEDURE pkg_adm_util.P1 AS
BEGIN
MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE ( current_schema ( ),'pkg_adm_util' ,'un_stand_value' ,
( 'A' ) ::TEXT );
MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE ( current_schema ( ),'pkg_adm_util' ,'un_stand_value' ,
( 'B' ) ::TEXT );
MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE ( current_schema ( ),'pkg_adm_util' ,'un_stand_value' ,
( 'C' ) ::TEXT );
DBMS_OUTPUT.PUT_LINE ( MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE( 'scott' ,'pkg_adm_util' ,'defaultdate' ) :: DATE );
DBMS_OUTPUT.PUT_LINE ( MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE('scott' ,'pkg_adm_util' ,'defaulttime' ) :: TIMESTAMP );
DBMS_OUTPUT.PUT_LINE ( MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE('scott' ,'pkg_adm_util' ,'un_stand_value' ) :: TEXT );
DBMS_OUTPUT.PUT_LINE ( MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE('scott' ,'pkg_adm_util' ,'un_stand_value' ) :: TEXT );
DBMS_OUTPUT.PUT_LINE ( MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE('scott' ,'pkg_adm_util' ,'un_stand_value' ) :: TEXT );
END;
/
If pkgSchemaNaming is true.

- Oracle supports package variables for multiple schemas. If different schemas have the same package and variable names, such as:
  - schema1.mypackage.myvariable
  - schema2.mypackage.myvariable

After migration, the schema names will not be used to differentiate the two package variables. Because schema names are ignored, the last data type declaration or operation for [any_schema].mypackage.myvariable will overwrite the type and value for schema1.mypackage.myvariable and schema2.mypackage.myvariable.

Input

- Package variable with default value declared in one package by using CONSTANT keyword and used in another package

By declaring global variable inside package specification, which can be accessed in same and other package as well.

```sql
PACKAGE "SAD"."BAS_SUBTYPE_PKG" : (Declaring global variable)
-------------------------------------------------
g_header_waiting_split_status CONSTANT VARCHAR2(20) := 'Waiting_Distribute';
PACKAGE SAD.sad_lookup_stage_pkg: (Used global variable)
--------------------------------------------------
PROCEDURE calc_product_price(pi_contract_no   IN VARCHAR2 DEFAULT NULL,
                                  pi_stage_id      IN NUMBER DEFAULT NULL,
                                  pi_calc_category IN VARCHAR2 DEFAULT 'all',
                                  pi_op_code       IN NUMBER,
                                  po_error_msg     OUT VARCHAR2)
IS
CURSOR cur_contract IS
    SELECT DISTINCT sdh.contract_number, sdh.stage_id
    FROM sad_distribution_headers_t sdh
    WHERE sdh.status = bas_subtype_pkg.g_header_waiting_split_status
    AND sdh.contract_number = nvl(pi_contract_no, sdh.contract_number)
    AND sdh.stage_id = nvl(pi_stage_id, sdh.stage_id);

v_ras_flag VARCHAR2 ( 1 ) ;
BEGIN
... END calc_product_price;
/
```

Output

```sql
PROCEDURE calc_product_price(pi_contract_no   IN VARCHAR2 DEFAULT NULL,
                                  pi_stage_id      IN NUMBER DEFAULT NULL,
                                  pi_calc_category IN VARCHAR2 DEFAULT 'all',
                                  pi_op_code       IN NUMBER,
                                  po_error_msg     OUT VARCHAR2)
IS
MIG_PV_VAL_DUMMY_G_HEADER_WAITING_SPLIT_STATUS VARCHAR2 ( 20 ) :=
MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE
( 'SAD', 'bas_subtype_pkg', 'g_header_waiting_split_status' ) ::VARCHAR2 ( 20 ) ;
CURSOR cur_contract IS
    SELECT DISTINCT sdh.contract_number, sdh.stage_id
    FROM sad_distribution_headers_t sdh
    WHERE sdh.status = MIG_PV_VAL_DUMMY_G_HEADER_WAITING_SPLIT_STATUS
    AND sdh.contract_number = nvl(pi_contract_no, sdh.contract_number)
    AND sdh.stage_id = nvl(pi_stage_id, sdh.stage_id);

v_ras_flag VARCHAR2 ( 1 ) ;
```
Package variable need to declare before CURSOR declaration.

**Input - Variable of type EXCEPTION**

Package variable is a kind of global variable, which can be used in entire package by declaring one time.

```sql
CREATE OR REPLACE PACKAGE BODY SAD.sad_lookup_stage_pkg IS
  ex_prog_error EXCEPTION;
PROCEDURE assert_null ( pi_value IN VARCHAR2 ) IS
BEGIN
  IF pi_value IS NOT NULL THEN
    RAISE ex_prog_error;
  END IF;
END assert_null;
END SAD.sad_lookup_stage_pkg
/
```

**Output**

```sql
CREATE OR REPLACE PROCEDURE SAD.sad_lookup_stage_pkg#assert_null
  ( pi_value IN VARCHAR2 )
PACKAGE
IS
  ex_prog_error EXCEPTION;
BEGIN
  IF pi_value IS NOT NULL THEN
    RAISE ex_prog_error;
  END IF;
END;
/
```

**NOTE**

As Gauss does not have package feature, package variable needs to declare in respective procedure or function where ever it is used.

**Input - If the configuration parameter "pkgSchemaNaming" is set to false**

Package variable is a kind of global variable, which can be used in entire package by declaring one time.

```sql
CREATE OR REPLACE PACKAGE BODY SAD.bas_lookup_misc_pkg IS
  g_pkg_name CONSTANT VARCHAR2(30) := 'bas_lookup_misc_pkg';
g_func_name VARCHAR2(30);
FUNCTION func_name RETURN VARCHAR2 IS
  l_func_name VARCHAR2(100);
BEGIN
  l_func_name := g_pkg_name || '.' || g_func_name;
END func_name;
END SAD.bas_lookup_misc_pkg
/
```
RETURN l_func_name;
END;
END SAD.bas_lookup_misc_pkg;
/

Output

INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES ( PACKAGE_NAME, SPEC_OR_BODY, VARIABLE_NAME, VARIABLE_TYPE, CONSTANT_I, DEFAULT_VALUE, RUNTIME_EXEC_I ) 
VALUES ( UPPER( 'bas_lookup_misc_pkg' ), 'B', UPPER( 'g_func_name' ), UPPER( 'VARCHAR2(30)' ), FALSE, NULL, FALSE ) ;
END ;
/

CREATE OR REPLACE FUNCTION SAD.bas_lookup_misc_pkg#func_name RETURN VARCHAR2
PACKAGE IS
  l_func_name VARCHAR2 ( 100 ) ;
  MIG_PV_VAL_DUMMY_G_PKG_NAME VARCHAR2 ( 30 ) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE ( 'SAD', 'bas_lookup_misc_pkg', 'g_pkg_name' ) ::VARCHAR2 ( 30 ) ;
  MIG_PV_VAL_DUMMY_G_FUNC_NAME VARCHAR2 ( 30 ) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE ( 'SAD', 'bas_lookup_misc_pkg', 'g_func_name' ) ::VARCHAR2 ( 30 ) ;
BEGIN
  l_func_name := MIG_PV_VAL_DUMMY_G_PKG_NAME || '.' || MIG_PV_VAL_DUMMY_G_FUNC_NAME ;
  MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE ( 'SAD', 'bas_lookup_misc_pkg', 'g_pkg_name', MIG_PV_VAL_DUMMY_G_PKG_NAME ) ;
  MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE ( 'SAD', 'bas_lookup_misc_pkg', 'g_func_name', MIG_PV_VAL_DUMMY_G_FUNC_NAME ) ;
  RETURN l_func_name ;
END ;
/

NOTE

If the configuration parameter "pkgSchemaNaming" is set to false, package variable migration is not happening properly in some places (that is GET to fetch default value and SET to assign final value are not adding).

However, this setting is not recommended by Kernel team. Please check with Kernel team.

Input-Package variable declared with data type as table column %TYPE

If data type is declared as table column %TYPE for a variable, it will consider the data type which is defined on table creation level for the respective column.

CREATE OR REPLACE PACKAGE BODY SAD.bas_lookup_misc_pkg IS
  v_emp_name emp.ename%TYPE;
  PROCEDURE save_emp_dtls ( v_empno IN VARCHAR2 ) ;
BEGIN
  IF v_emp_name IS NULL THEN
    v_emp_name := 'test';
  END IF;
END save_emp_dtls;
END bas_lookup_misc_pkg
/

Output

BEGIN
  INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES (
    PACKAGE_NAME,
    SPEC_OR_BODY,
    VARIABLE_NAME,
    VARIABLE_TYPE,
    CONSTANT_I,
    DEFAULT_VALUE,
    RUNTIME_EXEC_I
  )
  VALUES ( UPPER( 'bas_lookup_misc_pkg' ),
    'B',
    UPPER( 'v_emp_name' ),
    UPPER( 'VARCHAR2(30)' ),
    FALSE,
    NULL,
    FALSE ) ;
END ;
/
--*********************************************************
CREATE OR REPLACE PROCEDURE SAD.bas_lookup_misc_pkg#save_emp_dtls ( v_empno IN VARCHAR2 )
PACKAGE
IS
  MIG_PV_VAL_DUMMY_EMP_NAME VARCHAR2 ( 30 ) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE
    ( 'SAD', 'bas_lookup_misc_pkg' , 'v_emp_name' ) ::VARCHAR2 ( 30 ) ;
BEGIN
  IF MIG_PV_VAL_DUMMY_EMP_NAME IS NULL THEN
    MIG_PV_VAL_DUMMY_EMP_NAME := 'test';
  END IF ;
END ;
/

NOTE

While migrating package variable with data type as table column %TYPE, need to take the actual data type from table and need to use while declaring variable, not as %TYPE.

Input - If the configuration parameter "pkgSchemaNaming" is set to false

If PACKAGE name is specified along with SCHEMA name, need to use SCHEMA name on GET() to fetch default value and SET() to assign final value.

CREATE OR REPLACE PACKAGE BODY SAD.bas_lookup_misc_pkg IS
  g_pkg_name CONSTANT VARCHAR2(30) := 'bas_lookup_misc_pkg';
  g_func_name VARCHAR2(30);
  FUNCTION func_name RETURN VARCHAR2 IS
    l_func_name VARCHAR2(100);
    BEGIN
      l_func_name := g_pkg_name || '.' || g_func_name;
      RETURN l_func_name;
    END;
    END;
    END;
    END;
RETURN l_func_name;
END;
END SAD.bas_lookup_misc_pkg;
/

Output

BEGIN
  INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES ( PACKAGE_NAME, SPEC_OR_BODY, VARIABLE_NAME, VARIABLE_TYPE, CONSTANT_I, DEFAULT_VALUE, RUNTIME_EXEC_I ) VALUES ( UPPER('bas_lookup_misc_pkg'), 'B', UPPER('g_pkg_name'), UPPER('VARCHAR2(30)'), TRUE, 'bas_lookup_misc_pkg', FALSE );
  INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES ( PACKAGE_NAME, SPEC_OR_BODY, VARIABLE_NAME, VARIABLE_TYPE, CONSTANT_I, DEFAULT_VALUE, RUNTIME_EXEC_I ) VALUES ( UPPER('bas_lookup_misc_pkg'), 'B', UPPER('g_func_name'), UPPER('VARCHAR2(30)'), FALSE, NULL, FALSE );
END;
/

CREATE OR REPLACE FUNCTION SAD.bas_lookup_misc_pkg#func_name RETURN VARCHAR2
  PACKAGE
  IS
    l_func_name VARCHAR2 ( 100 ) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE ('SAD','bas_lookup_misc_pkg','g_func_name');
  BEGIN
    l_func_name := MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE ('SAD','bas_lookup_misc_pkg','g_pkg_name',MIG_PV_VAL_DUMMY_G_PKG_NAME);
Input - If the configuration parameter "pkgSchemaNaming" is set to false

If the configuration parameter "pkgSchemaNaming" is set to false.

```sql
CREATE OR REPLACE PACKAGE BODY bas_lookup_misc_pkg IS
    g_pkg_name CONSTANT VARCHAR2(30) := 'bas_lookup_misc_pkg';
    g_func_name VARCHAR2(30);

    FUNCTION func_name RETURN VARCHAR2 IS
        l_func_name VARCHAR2(100);
    BEGIN
        l_func_name := g_pkg_name || ':' || g_func_name;
        RETURN l_func_name;
    END;
END SAD.bas_lookup_misc_pkg;
/
```

Output

```sql
BEGIN
    INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES (
        PACKAGE_NAME, SPEC_OR_BODY, VARIABLE_NAME, VARIABLE_TYPE, CONSTANT_I, DEFAULT_VALUE, RUNTIME_EXEC_I
    ) VALUES ( UPPER('bas_lookup_misc_pkg'), 'B', UPPER('g_pkg_name'), UPPER('VARCHAR2(30)'), TRUE, 'bas_lookup_misc_pkg', FALSE ) ;
    INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES (
        PACKAGE_NAME, SPEC_OR_BODY, VARIABLE_NAME, VARIABLE_TYPE, CONSTANT_I, DEFAULT_VALUE, RUNTIME_EXEC_I
    ) VALUES ( UPPER('bas_lookup_misc_pkg'), 'B', UPPER('g_func_name'), UPPER('VARCHAR2(30)'), FALSE, NULL, FALSE ) ;
END ;
/
```

```sql
CREATE OR REPLACE FUNCTION bas_lookup_misc_pkg#func_name RETURN VARCHAR2 PACKAGE IS
    l_func_name VARCHAR2 ( 100 ) ;
BEGIN
    MIG_PV_VAL_DUMMY_G_PKG_NAME VARCHAR2 ( 30 ) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE ( CURRENT_SCHEMA() ,'bas_lookup_misc_pkg' ,["g_pkg_name"] ) ::VARCHAR2 ( 30 ) ;
    MIG_PV_VAL_DUMMY_G_FUNC_NAME VARCHAR2 ( 30 ) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE ( CURRENT_SCHEMA() ,'bas_lookup_misc_pkg' ,["g_func_name"] ) ::VARCHAR2 ( 30 ) ;
BEGIN
```
l_func_name := MIG_PV_VAL_DUMMY_G_PKG_NAME || '.' || MIG_PV_VAL_DUMMY_G_FUNC_NAME ;

MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE ( CURRENT_SCHEMA(),'bas_lookup_misc_pkg','g_pkg_name',MIG_PV_VAL_DUMMY_G_PKG_NAME ) ;
MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE ( CURRENT_SCHEMA(),'bas_lookup_misc_pkg','g_func_name',MIG_PV_VAL_DUMMY_G_FUNC_NAME ) ;

RETURN l_func_name ;
END ;
/

Input : if "pkgSchemaNaming" = false, package variable

The global variable is not correctly converted during package conversion, and an error is reported during compilation. If the configuration parameter "pkgSchemaNaming" is set to false, package variable migration is not happening properly in some places.

However, this setting is not recommended by Kernel team. It is recommended to check with Kernel team.

CREATE OR REPLACE PACKAGE BODY SAD.bas_dml_lookup_pkg IS
  g_pkg_name CONSTANT VARCHAR2(30) := 'bas_dml_ic_price_rule_pkg' ;
  g_func_name VARCHAR2 (100);

FUNCTION func_name
RETURNS VARCHAR2
IS
  l_func_name VARCHAR2(100) ;
BEGIN
  l_func_name := g_pkg_name || '.' || g_func_name ;
  RETURN l_func_name ;
END ;
END bas_dml_lookup_pkg ;
/

Output

BEGIN
  INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES ( USER_NAME, PACKAGE_NAME, SPEC_OR_BODY , VARIABLE_NAME, VARIABLE_TYPE , CONSTANT_I, DEFAULT_VALUE, RUNTIME_EXEC_I ) VALUES ( 'SAD', UPPER( 'bas_dml_lookup_pkg' ), 'B' , UPPER( 'g_pkg_name' ), UPPER( 'VARCHAR2 ( 30 )' ) , TRUE, 'bas_dml_ic_price_rule_pkg', FALSE ) ;
  INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES ( USER_NAME, PACKAGE_NAME, SPEC_OR_BODY , VARIABLE_NAME, VARIABLE_TYPE , CONSTANT_I, DEFAULT_VALUE, RUNTIME_EXEC_I ) VALUES ( 'SAD', UPPER( 'bas_dml_lookup_pkg' ), 'B' , UPPER( 'g_func_name' ), UPPER( 'VARCHAR2(100)' ) , FALSE, NULL, FALSE ) ;
END ;
/

CREATE OR REPLACE FUNCTION SAD.bas_dml_lookup_pkg#func_name
RETURNS VARCHAR2
IS
  MIG_PV_VAL_DUMMY_G_PKG_NAME VARCHAR2(30) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE
CREATE OR REPLACE FUNCTION SAD.bas_dml_lookup_pkg#func_name
RETURN VARCHAR2
IS
MIG_PV_VAL_DUMMY_G_PKG_NAME VARCHAR2(30) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE
( 'SAD', 'BAS_DML_LOOKUP_PKG', 'G_PKG_NAME' )::VARCHAR2(30) ;
MIG_PV_VAL_DUMMY_G_FUNC_NAME VARCHAR2(100) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE
( 'SAD', 'BAS_DML_LOOKUP_PKG', 'G_FUNC_NAME' )::VARCHAR2(100) ;
l_func_name VARCHAR2(100) ;
BEGIN
  l_func_name := MIG_PV_VAL_DUMMY_G_PKG_NAME || '.' || MIG_PV_VAL_DUMMY_G_FUNC_NAME ;
  RETURN l_func_name ;
END ;
/

Input: table field type definition in the (%type) table

During package conversion, the schema definition is not added to the table field type definition in the (%type) table. An error is reported during compilation.

CREATE TABLE CTP_BRANCH
( ID VARCHAR2(10) , NAME VARCHAR2(100) , DESCRIPTION VARCHAR2(500) );

CREATE OR REPLACE PACKAGE BODY SAD.bas_dml_lookup_pkg IS
  g_pkg_name CONSTANT VARCHAR2(30) := 'bas_dml_ic_price_rule_pkg' ;
  g_func_name CTP_BRANCH.NAME%TYPE;

FUNCTION func_name
RETURN VARCHAR2
IS
  l_func_name VARCHAR2(100) ;
BEGIN
  l_func_name := g_pkg_name || '.' || g_func_name ;
  RETURN l_func_name ;
END ;
END bas_dml_lookup_pkg ;
/

Output

BEGIN
  INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES (
    USER_NAME, PACKAGE_NAME, SPEC_OR_BODY , VARIABLE_NAME, VARIABLE_TYPE , CONSTANT_I, DEFAULT_VALUE, RUNTIME_EXEC_I
  )
  VALUES ( 'SAD', UPPER( 'bas_dml_lookup_pkg' ), 'B' , UPPER( 'g_pkg_name' ) , UPPER( 'VARCHAR2 ( 30 )' ) , TRUE, 'bas_dml_ic_price_rule_pkg', FALSE ) ;

  INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES (
    USER_NAME, PACKAGE_NAME, SPEC_OR_BODY , VARIABLE_NAME, VARIABLE_TYPE , CONSTANT_I, DEFAULT_VALUE, RUNTIME_EXEC_I
  )
  VALUES ( 'SAD', UPPER( 'bas_dml_lookup_pkg' ), 'B' , UPPER( 'g_func_name' ) , UPPER( 'VARCHAR2(100)' ) , FALSE, NULL, FALSE ) ;

END ;
/

CREATE OR REPLACE FUNCTION SAD.bas_dml_lookup_pkg#func_name
RETURN VARCHAR2
IS
MIG_PV_VAL_DUMMY_G_PKG_NAME VARCHAR2(30) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE
( 'SAD', 'BAS_DML_LOOKUP_PKG', 'G_PKG_NAME' )::VARCHAR2(30) ;
MIG_PV_VAL_DUMMY_G_FUNC_NAME VARCHAR2(100) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE
( 'SAD', 'BAS_DML_LOOKUP_PKG', 'G_FUNC_NAME' )::VARCHAR2(100) ;
l_func_name VARCHAR2(100) ;
BEGIN
  l_func_name := MIG_PV_VAL_DUMMY_G_PKG_NAME || '.' || MIG_PV_VAL_DUMMY_G_FUNC_NAME;
  RETURN l_func_name;
END;
/

Exception

Package variable can be declared as EXCEPTION, which is not supported in Gauss.

Input

CREATE OR REPLACE PACKAGE BODY product_pkg IS
  ex_prog_error EXCEPTION;
  PROCEDURE assert_null(pi_value IN VARCHAR2) IS
    BEGIN
      IF pi_value IS NOT NULL THEN
        RAISE ex_prog_error;
      END IF;
    EXCEPTION
      WHEN ex_prog_error THEN
        RAISE ex_prog_error;
    END assert_null;
END product_pkg;
/

Output

CREATE OR REPLACE PROCEDURE product_pkg.Assert_null (pi_value IN VARCHAR2) IS
  ex_prog_error EXCEPTION;
  BEGIN
    IF pi_value IS NOT NULL THEN
      RAISE ex_prog_error;
    END IF;
  EXCEPTION
    WHEN ex_prog_error THEN
      RAISE ex_prog_error;
  END;
/

Default Value

function is specified as a default value for a package variable.

Input

BEGIN
  INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES (PACKAGE_NAME, SPEC_OR_BODY, VARIABLE_NAME, VARIABLE_TYPE, CONSTANT_I, DEFAULT_VALUE, RUNTIME_EXEC_I)
  VALUES (UPPER('PKG_REVN_ARPU'), 'B', UPPER('imodel'), UPPER('log_table.ds_exec%TYPE'), FALSE, pkg_etl.proc_set_chain('DAILY ARPU'))
CREATE OR REPLACE PACKAGE BODY IC_STAGE.PKG_REVN_ARPU AS
  imodel   log_table.ds_exec%TYPE := pkg_etl.proc_set_chain ('DAILY ARPU');
PROCEDURE AGGR_X_AGG00_REVN_DEALER (p_date    PLS_INTEGER,
  p_days    PLS_INTEGER)
AS
  v_start_date   PLS_INTEGER;
  v_curr_date    PLS_INTEGER;
  v_imodel   VARCHAR2(100);
BEGIN
  pkg_etl.proc_start (p_date, 'AGGR_X_AGG00_REVN_DEALER ');
  v_start_date := TO_CHAR (TO_DATE (p_date, 'yyyymmdd') - (p_days - 1), 'yyyymmdd');
  v_curr_date := p_date;
  v_imodel := imodel;
END;
END PKG_REVN_ARPU;
/

Output

BEGIN
  INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES (
    PACKAGE_NAME
    ,SPEC_OR_BODY
    ,VARIABLE_NAME
    ,VARIABLE_TYPE
    ,CONSTANT_I
    ,DEFAULT_VALUE
    ,RUNTIME_EXEC_I
  )
VALUES ( UPPER( 'PKG_REVN_ARPU' )
  ,B
  ,UPPER( 'imodel' )
  ,UPPER( 'log_table.ds_exec%TYPE' )
  ,FALSE
  ,$q$pkg_etl.proc_set_chain ('DAILY ARPU')$q$
  ,TRUE ) ;
END ;
/
CREATE OR REPLACE PROCEDURE PKG_REVN_ARPU.AGGR_X_AGG00_REVN_DEALER ( p_date INTEGER
  ,p_days INTEGER )
AS
  MIG_PV_VAL_DUMMY_IMODEL log_table.ds_exec%TYPE := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE
  ( CURRENT_USER,'PKG_REVN_ARPU','imodel' ) ::log_table.ds_exec%TYPE ;
  v_start_date INTEGER ;
  v_curr_date INTEGER ;
  v_imodel VARCHAR2 ( 100 ) ;
BEGIN
  pkg_etl.proc_start ( p_date , 'AGGR_X_AGG00_REVN_DEALER ' ) ;
  v_start_date := TO_CHAR( TO_DATE( p_date , 'yyyymmdd' ) - ( p_days - 1 ) , 'yyyymmdd' );
  v_curr_date := p_date;
  v_imodel := imodel;
END;
PLS_INTEGER

PLS_INTEGER datatype not converted into INTEGER for package variables but it is working fine for other local variables therefore for package variables also PLS_INTEGER should be converted to INTEGER datatype that is variable1 PLS_INTEGER => variable1 INTEGER

SCRIPTS : SAD_CALC_BPART_PRICE_PKG.sql, SAD_CALC_ITEM_PKG_TEST_OB.sql, SAD_CALC_ITEM_PRICE_TEST_OB.sql, SAD_CALC_ITEM_PRI_TEST_OB.sql, SAD_CALC_ITEM_TEST_OB.sql

INPUT :

CREATE OR REPLACE PACKAGE BODY "SAD"."SAD_CALC_BPART_PRICE_PKG" IS
g_max_number_of_entities PLS_INTEGER := 100;
FUNCTION split_warning(pi_contract_number IN VARCHAR2,
pi_stage_id IN NUMBER,
pi_quotation_id IN NUMBER,
p_cfg_instance_id IN NUMBER) RETURN VARCHAR2 IS
BEGIN
---
l_item_list := items_no_cost(pi_contract_number => pi_contract_number,
pi_stage_id => pi_stage_id,
pi_quotation_id => pi_quotation_id,
pi_cfg_instance_id => pi_cfg_instance_id,
pi_max_number_of_entities => g_max_number_of_entities,
pi_item_sep_char => g_item_sep_char,
po_error_msg => po_error_msg);
---
END split_warning;
END SAD_CALC_BPART_PRICE_PKG;

Output

CREATE OR REPLACE FUNCTION SAD.SAD_CALC_BPART_PRICE_PKG#split_warning ( pi_contract_number IN VARCHAR2,
pi_stage_id IN NUMBER
pi_quotation_id IN NUMBER
pi_cfg_instance_id IN NUMBER ) RETURN VARCHAR2 IS
---
MIG_PV_VAL_DUMMY_G_MAX_NUMBER_OF_ENTITIES PLS_INTEGER :=
MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE ( current_schema ( )
'SAD_CALC_BPART_PRICE_PKG',
'g_max_number_of_entities' ) ::PLS_INTEGER ;
---
l_item_list := SAD.SAD_CALC_BPART_PRICE_PKG#items_no_cost ( pi_contract_number =>
pi_stage_id => pi_stage_id,
pi_quotation_id => pi_quotation_id,
pi_cfg_instance_id => pi_cfg_instance_id,
pi_max_number_of_entities => MIG_PV_VAL_DUMMY_G_MAX_NUMBER_OF_ENTITIES,
pi_item_sep_char => MIG_PV_VAL_DUMMY_G_ITEM_SEP_CHAR,
po_error_msg => po_error_msg ) ;
---
END;

PLS_INTEGER datatype not converted into INTEGER for package variables but it is working fine for other local variables therefore for package variables also
PLSTEGER should be converted to INTEGER datatype that is varaible1
PLSTEGER => varaible1 INTEGER.

SCRIPTS: SAD_CALC_BPART_PRICE_PKG.sql, SAD_CALC_ITEM_PKG_TEST_OB.sql,
SAD_CALC_ITEM_PRICE_TEST_OB.sql, SAD_CALC_ITEM_PRI_TEST_OB.sql,
SAD_CALC_ITEM_TEST_OB.sql

Input

CREATE OR REPLACE PACKAGE BODY "SAD"."SAD_CALC_BPART_PRICE_PKG" IS

  g_max_number_of_entities PLS_INTEGER := 100;

  FUNCTION split_warning(pi_contract_number IN VARCHAR2,
                          pi_stage_id        IN NUMBER,
                          pi_quotation_id    IN NUMBER,
                          pi_cfg_instance_id IN NUMBER) RETURN VARCHAR2 IS
    BEGIN
      ---
      l_item_list := items_no_cost(pi_contract_number        => pi_contract_number,
                                    pi_stage_id               => pi_stage_id,
                                    pi_quotation_id           => pi_quotation_id,
                                    pi_cfg_instance_id        => pi_cfg_instance_id,
                                    pi_max_number_of_entities => g_max_number_of_entities,
                                    pi_sep_char               => g_item_sep_char,
                                    po_error_msg              => po_error_msg);
      ---

      END split_warning;

END SAD_CALC_BPART_PRICE_PKG;

Output

BEGIN
  INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES
  ( PACKAGE_NAME, SPEC_OR_BODY, VARIABLE_NAME,
    VARIABLE_TYPE, CONSTANT_I, DEFAULT_VALUE,
    RUNTIME_EXEC_I )
  VALUES ( UPPER( 'SAD_CALC_BPART_PRICE_PKG' ), 'B', UPPER( 'g_max_number_of_entities' ),
                       UPPER( 'INTEGER' ), FALSE, 100, FALSE);
END;
/

CREATE OR REPLACE FUNCTION SAD.SAD_CALC_BPART_PRICE_PKG#split_warning
( pi_contract_number IN VARCHAR2,
  pi_stage_id   IN NUMBER )
RETURN VARCHAR2
PACKAGE
IS
  MIG_PV_VAL_DUMMY_G_MAX_NUMBER_OF_ENTITIES INTEGER :=
    MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE( 'SAD', 'SAD_CALC_BPART_PRICE_PKG',
    'g_max_number_of_entities' ) :: INTEGER;
  po_error_msg sad_products_t.exception_description%TYPE ;

BEGIN
  l_item_list := items_no_cost( pi_contract_number => pi_contract_number,
                                pi_stage_id => pi_stage_id,
                                pi_max_number_of_entities => MIG_PV_VAL_DUMMY_G_MAX_NUMBER_OF_ENTITIES,
                                po_error_msg => po_error_msg);
  MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE( 'SAD', 'SAD_CALC_BPART_PRICE_PKG',
  'g_max_number_of_entities', MIG_PV_VAL_DUMMY_G_MAX_NUMBER_OF_ENTITIES);
END;
RETURN po_error_msg;

EXCEPTION
  WHEN OTHERS THEN
    po_error_msg := 'Program Others abnormal, Fail to obtain the warning information.' || SQLERRM;
    MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE('SAD', 'SAD_CALC_BPART_PRICE_PKG', 'g_max_number_of_entities', MIG_PV_VAL_DUMMY_G_MAX_NUMBER_OF_ENTITIES);
    RETURN po_error_msg;
END;
/

Cursor With Package Variable

The cursor declared in SAD.sad_calc_product_price_pkg#calc_product_price contains package variables and needs to be addressed.

Input

CREATE OR REPLACE PACKAGE SAD.bas_subtype_pkg IS
  g_header_waiting_split_status CONSTANT VARCHAR2(20) := 'Waiting_Distribute';
  SUBTYPE error_msg IS sad_products_t.exception_description%TYPE;
END bas_subtype_pkg;
/

CREATE OR REPLACE PACKAGE BODY SAD.sad_calc_product_price_pkg IS
  PROCEDURE calc_product_price(pi_contract_no IN VARCHAR2 DEFAULT NULL,
  pi_stage_id IN NUMBER DEFAULT NULL,
  po_error_msg OUT VARCHAR2) IS
    CURSOR cur_contract IS
      SELECT DISTINCT sdh.contract_number, sdh.stage_id
      FROM sad_distribution_headers_t sdh
      WHERE sdh.status = bas_subtype_pkg.g_header_waiting_split_status
      AND sdh.contract_number = nvl(pi_contract_no, sdh.contract_number)
      AND sdh.stage_id = nvl(pi_stage_id, sdh.stage_id);
    lv_error_msg bas_subtype_pkg.error_msg;
    BEGIN
      FOR rec_contract IN cur_contract LOOP
        validate_process_status(rec_contract.contract_number,
                                rec_contract.stage_id,
                                lv_error_msg);
      END LOOP;
      po_error_msg := lv_error_msg;
    END calc_product_price;
END sad_calc_product_price_pkg;
/

Output

BEGIN
  INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES
  ( PACKAGE_NAME, SPEC_OR_BODY, VARIABLE_NAME, VARIABLE_TYPE, CONSTANT_I, DEFAULT_VALUE, RUNTIME_EXEC_I )
  VALUES ( 'SAD', 'SAD_CALC_BPART_PRICE_PKG', 'g_header_waiting_split_status', 'VARCHAR2(20)', TRUE, 'Waiting_Distribute' , FALSE );
END;
/
CREATE OR REPLACE PROCEDURE SAD.sad_calc_product_price_pkg#calc_product_price
( pi_contract_no IN VARCHAR2 DEFAULT NULL
 , pi_stage_id IN NUMBER DEFAULT NULL
 , po_error_msg OUT VARCHAR2 )
PACKAGE
IS
 MIG_PV_VAL_DUMMY_G_HEADER_WAITING_SPLIT_STATUS VARCHAR2 ( 20 ) :=
 MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE ( 'SAD' ,'bas_subtype_pkg'
 , 'g_header_waiting_split_status' ) ::VARCHAR2 ( 20 ) ;
CURSOR cur_contract IS
SELECT DISTINCT sdh.contract_number, sdh.stage_id
FROM sad_distribution_headers_t sdh
WHERE sdh.status = MIG_PV_VAL_DUMMY_G_HEADER_WAITING_SPLIT_STATUS
AND sdh.contract_number = nvl( pi_contract_no ,sdh.contract_number )
AND sdh.stage_id = nvl( pi_stage_id ,sdh.stage_id ) ;
lv_error_msg sad_products_t.exception_description%TYPE ;
BEGIN
 FOR rec_contract IN cur_contract
 LOOP
 validate_process_status ( rec_contract.contract_number ,rec_contract.stage_id ,lv_error_msg ) ;
 END LOOP ;
 po_error_msg := lv_error_msg ;
 MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE
 ( 'SAD' ,'bas_subtype_pkg' ,'g_header_waiting_split_status' ,MIG_PV_VAL_DUMMY_G_HEADER_WAITING_SPLIT_STATUS ) ;
END ;
/

SET VARIABLE function after the RETURN

SET VARIABLE function should be called before the RETURN statements in procedure and function.

Input

CREATE OR REPLACE PACKAGE BODY SAD.bas_dml_lookup_pkg IS
 g_pkg_name CONSTANT VARCHAR2(30) := 'bas_dml_lookup_pkg' ;
 g_func_name VARCHAR2(100) ;
 FUNCTION func_name
 RETURN VARCHAR2
 IS
 l_func_name VARCHAR2(100) ;
 BEGIN
 g_func_name := 'func_name';
 l_func_name := g_pkg_name || ':' || g_func_name ;
 RETURN l_func_name ;
 END;
 PROCEDURE data_change_logs ( pi_table_name IN VARCHAR2
 , pi_table_key_columns IN VARCHAR2
 , po_error_msg OUT VARCHAR2 
 )
 IS 
 BEGIN
 g_func_name := 'data_change_logs';
 IF pi_table_name IS NULL THEN
 RETURN;
 END IF;
 INSERT INTO fnd_data_change_logs_t
 ( logid, table_name, table_key_columns )

VALUES ( fnd_data_change_logs_t_s.NEXTVAL, pi_table_name, pi_table_key_columns );
EXCEPTION
WHEN OTHERS THEN
  po_error_msg := 'Others Exception raise in ' || func_name || ',' || SQLERRM;
END data_change_logs;
END bas_dml_lookup_pkg;
/

Output

BEGIN
  INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES ( PACKAGE_NAME, SPEC_OR_BODY, VARIABLE_NAME, VARIABLE_TYPE, CONSTANT_I, DEFAULT_VALUE, RUNTIME_EXEC_I )
  VALUES ( UPPER('bas_dml_lookup_pkg'), 'B', UPPER('g_pkg_name'), UPPER('VARCHAR2(30) '), TRUE, 'bas_dml_lookup_pkg', FALSE );

  INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES ( PACKAGE_NAME, SPEC_OR_BODY, VARIABLE_NAME, VARIABLE_TYPE, CONSTANT_I, DEFAULT_VALUE, RUNTIME_EXEC_I )
  VALUES ( UPPER('bas_dml_lookup_pkg'), 'B', UPPER('g_func_name'), UPPER('VARCHAR2(100) '), FALSE, NULL, FALSE );

END;
/
CREATE OR REPLACE FUNCTION SAD.bas_dml_lookup_pkg#func_name
BEGIN
  MIG_PV_VAL_DUMMY_G_PKG_NAME VARCHAR2 ( 30 ) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE ( 'SAD', 'bas_dml_lookup_pkg', 'g_pkg_name' ) ::VARCHAR2 ( 30 ) ;
  MIG_PV_VAL_DUMMY_G_FUNC_NAME VARCHAR2 ( 100 ) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE ( 'SAD', 'bas_dml_lookup_pkg', 'g_func_name' ) ::VARCHAR2 ( 100 ) ;
  l_func_name VARCHAR2 ( 100 ) ;
  BEGIN
    MIG_PV_VAL_DUMMY_G_FUNC_NAME := 'func_name';
    l_func_name := MIG_PV_VAL_DUMMY_G_PKG_NAME || '.' || MIG_PV_VAL_DUMMY_G_FUNC_NAME ;
    MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE ( 'SAD', 'bas_dml_lookup_pkg', 'func_name' ) :=VARCHAR2 ( 100 ) ;
  RETURN l_func_name ;
END;
/
CREATE OR REPLACE PROCEDURE SAD.bas_dml_lookup_pkg#data_change_logs ( pi_table_name IN VARCHAR2, pi_table_key_columns IN VARCHAR2 , po_error_msg OUT VARCHAR2 )
BEGIN
  MIG_PV_VAL_DUMMY_G_PKG_NAME VARCHAR2 ( 30 ) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE ( 'SAD', 'bas_dml_lookup_pkg', 'g_pkg_name' ) ::VARCHAR2 ( 30 ) ;
  MIG_PV_VAL_DUMMY_G_FUNC_NAME VARCHAR2 ( 100 ) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE ( 'SAD', 'bas_dml_lookup_pkg', 'g_func_name' ) ::VARCHAR2 ( 100 ) ;
  IF pi_table_name IS NULL THEN
    MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE ( 'SAD', 'bas_dml_lookup_pkg', 'data_change_logs' ) ;
  RETURN ;
END IF ;
INSERT INTO fnd_data_change_logs_t (logid, table_name, table_key_columns) VALUES (NEXTVAL ( 'fnd_data_change_logs_t_s' ), pi_table_name, pi_table_key_columns);

MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE ('SAD' , 'bas_dml_lookup_pkg' , 'g_func_name' , MIG_PV_VAL_DUMMY_G_FUNC_NAME ) ;

EXCEPTION
WHEN OTHERS THEN
    po_error_msg := 'Others Exception raise in ' || SAD.bas_dml_lookup_pkg#func_name ( ) || ',' || SQLERRM ;
MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE ('SAD' , 'bas_dml_lookup_pkg' , 'g_func_name' , MIG_PV_VAL_DUMMY_G_FUNC_NAME ) ;
END ;
/

Empty Package

No need to migrate an empty package body.

Input
CREATE OR REPLACE PACKAGE BODY SAD.bas_subtype_pkg IS
BEGIN
    NULL;
END bas_subtype_pkg;
/

Output will be an empty file.

6.13.11.4 Package Split

Package spec is migrated as create schema name with the package name and the procedures and functions in Package body is migrated as Packagename.procedurename Packagename.functionname.

It depends on the following configuration item pkgSchemaNaming=true

Input – PACKAGE1.FUNC1

CREATE OR REPLACE PACKAGE BODY pack AS
FUNCTION get_fullname(n_emp_id NUMBER) RETURN VARCHAR2 IS
    v_fullname VARCHAR2(46);
BEGIN
    SELECT first_name || ',' || last_name INTO v_fullname FROM employees WHERE employee_id = n_emp_id;
    RETURN v_fullname;
END get_fullname;
PROCEDURE get_salary(n_emp_id NUMBER) RETURN NUMBER IS
    n_salary NUMBER(8,2);
BEGIN
    SELECT salary INTO n_salary FROM employees WHERE employee_id = n_emp_id;
    END get_salary;
END pack;
/

Output
CREATE OR REPLACE FUNCTION pack.get_fullname ( n_emp_id NUMBER ) 
RETURN VARCHAR2 IS v_fullname VARCHAR2 ( 46 ) ;
BEGIN
SELECT
    first_name || ',' || last_name INTO v_fullname
FROM
    employees
WHERE
    employee_id = n_emp_id ;
RETURN v_fullname ;
END ;
/

CREATE OR REPLACE PROCEDURE pack.get_salary ( n_emp_id NUMBER ) 
RETURN NUMBER IS n_salary NUMBER ( 8 ,2 ) ;
BEGIN
SELECT
    salary INTO n_salary
FROM
    employees
WHERE
    employee_id = n_emp_id ;
END ;
/

if "pkgSchemaNaming" = false, package split
The package invoking in the package is not converted.
bas_lookup_misc_pkg.insert_fnd_data_change_logs ( 

Input
CREATE OR REPLACE PACKAGE BODY SAD.bas_dml_lookup_pkg IS
    g_pkg_name CONSTANT VARCHAR2(30) := 'bas_dml_ic_price_rule_pkg' ;
    g_func_name VARCHAR2(100);
FUNCTION func_name
RETURN VARCHAR2
IS
    l_func_name VARCHAR2(100) ;
    BEGIN
        l_func_name := g_pkg_name || '.' || g_func_name ;
        RETURN l_func_name ;
    END ;
PROCEDURE data_change_logs ( pi_table_name        IN VARCHAR2 , pi_table_key_columns IN VARCHAR2 , po_error_msg         OUT VARCHAR2 )
IS
BEGIN
  g_func_name := 'insert_fnd_data_change_logs_t';

  INSERT INTO fnd_data_change_logs_t
    ( logid, table_name, table_key_columns )
  VALUES
    ( fnd_data_change_logs_t_s.NEXTVAL,
      pi_table_name, pi_table_key_columns );
EXCEPTION
  WHEN OTHERS THEN
    po_error_msg := 'Others Exception raise in ' || func_name || ',' || SQLERRM;
END data_change_logs;
END bas_dml_lookup_pkg;
/

CREATE OR REPLACE FUNCTION SAD.bas_dml_lookup_pkg#func_name
RETURN VARCHAR2
IS
  MIG_PV_VAL_DUMMY_G_PKG_NAME VARCHAR2(30) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE
    ( 'SAD', 'BAS_DML_LOOKUP_PKG', 'G_PKG_NAME' )::VARCHAR2(30) ;
  MIG_PV_VAL_DUMMY_G_FUNC_NAME VARCHAR2(100) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE
    ( 'SAD', 'BAS_DML_LOOKUP_PKG', 'G_FUNC_NAME' )::VARCHAR2(100) ;
l_func_name VARCHAR2(100) ;
BEGIN
  l_func_name := MIG_PV_VAL_DUMMY_G_PKG_NAME || '.' || MIG_PV_VAL_DUMMY_G_FUNC_NAME ;
  RETURN l_func_name ;
END ;
/
CREATE OR REPLACE PROCEDURE SAD.bas_dml_lookup_pkg#data_change_logs ( pi_table_name IN VARCHAR2,
pi_table_key_columns IN VARCHAR2, po_error_msg OUT VARCHAR2 )
IS
  MIG_PV_VAL_DUMMY_G_FUNC_NAME VARCHAR2(30) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE
    ( 'SAD', 'BAS_DML_LOOKUP_PKG', 'G_FUNC_NAME' )::VARCHAR2(30) ;
BEGIN
  MIG_PV_VAL_DUMMY_G_FUNC_NAME := 'insert_fnd_data_change_logs_t' ;
  INSERT INTO fnd_data_change_logs_t
    ( logid,table_name,table_key_columns )
  VALUES
    ( NEXTVAL ( 'fnd_data_change_logs_t_s' ), pi_table_name, pi_table_key_columns );
  MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE ( 'SAD', 'BAS_DML_LOOKUP_PKG', 'G_FUNC_NAME',
    MIG_PV_VAL_DUMMY_G_FUNC_NAME ) ;
EXCEPTION
  WHEN OTHERS THEN
    po_error_msg := 'Others Exception raise in ' || SAD.bas_dml_lookup_pkg#func_name() || ',' || SQLERRM ;
  MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE ( 'SAD', 'BAS_DML_LOOKUP_PKG', 'G_FUNC_NAME',
    MIG_PV_VAL_DUMMY_G_FUNC_NAME ) ;
END ;
/

PACKAGE Keyword

The kernel needs to add the package tag to the functions and stored procedures converted from the Package.

Input
CREATE OR REPLACE PACKAGE BODY SAD.bas_dml_lookup_pkg IS
FUNCTION func_name  
RETURN VARCHAR2  
IS  
    l_func_name VARCHAR2(100) ;  
BEGIN  
    l_func_name := 'bas_dml_lookup_pkg' || '.' || 'func_name' ;  
    RETURN l_func_name ;  
END ;  
END bas_dml_lookup_pkg ; /

CREATE OR REPLACE FUNCTION func_name  
RETURN VARCHAR2  
PACKAGE  
IS  
    l_func_name VARCHAR2(100) ;  
BEGIN  
    l_func_name := 'bas_dml_lookup_pkg' || '.' || 'func_name' ;  
    RETURN l_func_name ;  
END ;  
END bas_dml_lookup_pkg ; /

6.13.11.5 REF CURSOR

A REF Cursor is a datatype that holds the database cursor values and is used to return query results. Migration tool supports migration of REF CURSOR. The following example shows how the tool migrates lref_strong_emptyp (Local REF CURSOR) and ref_strong_emptyp (Package Level REF CURSOR).

**Input - REF CURSOR in PL/SQL Package** (Package Specification and Body)

```sql
# Package specification
CREATE OR REPLACE PACKAGE pkg_refcur
IS  
    TYPE ref_variable IS REF CURSOR;
    TYPE ref_strong_emptyp IS REF CURSOR RETURN emp_o%ROWTYPE;
    PROCEDURE p_get_employees ( v_id in INTEGER , po_results OUT ref_strong_emptyp );
END pkg_refcur ; /
# Package body
CREATE OR REPLACE PACKAGE BODY pkg_refcur
IS  
    TYPE lref_strong_emptyp IS REF CURSOR RETURN emp_o%ROWTYPE ;
    var_num NUMBER ;
    PROCEDURE p_get_employees ( v_id IN INTEGER , po_results OUT ref_strong_emptyp )
    is  
        vemp_rc lref_strong_emptyp ;
    Begin  
        OPEN po_results for  
            SELECT * FROM emp_o e  
            WHERE e.id = v_id ;
    EXCEPTION  
        WHEN OTHERS THEN  
            RAISE;
    END p_get_employees;
END pkg_refcur ; /
```

**Output**

6 Migration Tool - SQL Syntax Migration Tool
BEGIN
    INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES
    ( SCHEMA_NAME ,PACKAGE_NAME ,SPEC_OR_BODY ,VARIABLE_NAME ,VARIABLE_TYPE ,CONSTANT_I ,DEFAULT_VALUE ,EXPRESSION_I )
    VALUES ( UPPER( current_schema ) ,UPPER( 'pkg_refcur' ) ,'B' ,UPPER( 'var_num' ) ,UPPER( 'NUMBER' ) ,false ,NULL ,false ) ;
END ;
/

CREATE OR REPLACE PROCEDURE pkg_refcur#p_get_employees ( v_id IN INTEGER ,po_results OUT SYS_REFCURSOR ) is vemp_rc SYS_REFCURSOR ;
Begin
    OPEN po_results for SELECT
      *
    FROM
      emp_o e
    WHERE
      e.id = v_id ;
    EXCEPTION WHEN OTHERS
      THEN RAISE ;
END ;
/

6.13.11.6 VARRAY

This is REF CURSOR defined as return parameter.
It depends on the following configuration item:
plsqlCollection=varray

Input - VARRAY

CREATE OR REPLACE TYPE TYPE_RMTS_ARRAYTYPE IS TABLE OF VARCHAR2 (30000);

CREATE OR REPLACE PACKAGE BODY SCMS_STRING_UTILS
As
FUNCTION END_WITH (SRCSTRING VARCHAR2, --Source character string ENDCHAR VARCHAR2, --End character string IGNORECASE BOOLEAN --Ignore Case )
RETURN BOOLEAN IS SRCLEN NUMBER (20) := LENGTH(SRCSTRING);
ENDLEN NUMBER (20) := LENGTH(ENDCHAR);
V_TOKEN_ARRAY TYPE_RMTS_ARRAYTYPE := TYPE_RMTS_ARRAYTYPE ();
V_TOKEN_ARRAY1 TYPE_RMTS_ARRAYTYPE := TYPE_RMTS_ARRAYTYPE ();
I NUMBER (20) := 1;
TMP_CHAR VARCHAR(1);
TMP_CHAR1 VARCHAR(1);
BEGIN...
END;
END;
/

Output

CREATE OR REPLACE FUNCTION SCMS_STRING_UTILS.END_WITH (SRCSTRING VARCHAR2 /* source character string */ , ENDCHAR VARCHAR2 /* End character string */ , IGNORECASE BOOLEAN /* Ignore case */ )

6.13.11.7 Creating a Schema for Package

Package specification is migrated as create schema with that package name and the schema is not considered. It depends on the following configuration item:

pkgSchemaNaming=false.

Input – Create schema for Package

CREATE OR REPLACE EDITIONABLE PACKAGE "PACK_DEMO"."PACKAGE_GET_NOVA_INFO" AS

    TYPE novalistcur is REF CURSOR;

    PROCEDURE getNovalInfo (
        i_appEnShortName IN VARCHAR2,
        o_flag OUT VARCHAR2,
        o_errormsg OUT VARCHAR2,
        o_novalist OUT novalistcur
    );

Output

/*~~PACKAGE_GET_NOVA_INFO~~*/
CREATE
    SCHEMA PACKAGE_GET_NOVA_INFO

6.13.11.8 Grant Execute

This feature is used to give privileges to a specific user for a specific package. As Gauss does not support packages, all the procedures and functions defined in the specific package will be granted the execution permission.

Input

GRANT EXECUTE ON SAD.BAS_LOOKUP_MISC_PKG TO EIP_SAD;

Output

GRANT EXECUTE ON procedure_name TO EIP_SAD;
GRANT EXECUTE ON function1_name TO EIP_SAD;

Note

Here, procedure _name and function1_name both belong to SAD.BAS_LOOKUP_MISC_PKG.

Grant exec on Package

The last authorization of the package is not converted.

--GRANT

Input
---

Below should be created as `spec/t603.sql`

```sql
CREATE OR REPLACE PACKAGE SAD.bas_dml_lookup_pkg IS
  FUNCTION func_name RETURN VARCHAR2;
  PROCEDURE data_change_logs ( pi_table_name IN VARCHAR2,
                                pi_table_key_columns IN VARCHAR2,
                                po_error_msg OUT VARCHAR2 );
END bas_dml_lookup_pkg;
/
GRANT EXECUTE ON SAD.bas_dml_lookup_pkg TO eip_sad;
```

---

Below should be created as `body/t603.sql`

```sql
CREATE OR REPLACE PACKAGE BODY SAD.bas_dml_lookup_pkg IS
  g_pkg_name CONSTANT VARCHAR2(30) := 'bas_dml_ic_price_rule_pkg';
  g_func_name VARCHAR2(100);
  FUNCTION func_name
    RETURN VARCHAR2
  IS
    l_func_name VARCHAR2(100) := g_pkg_name || '.' || g_func_name;
    RETURN l_func_name;
  END func_name;
  PROCEDURE data_change_logs ( pi_table_name IN VARCHAR2,
                               pi_table_key_columns IN VARCHAR2,
                               po_error_msg OUT VARCHAR2 )
  IS
    BEGIN
    ...
  END data_change_logs;
END bas_dml_lookup_pkg;
/
```

---

### Output

```sql
BEGIN
  INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES
  ( PACKAGE_NAME,SPEC_OR_BODY,VARIABLE_NAME,
    VARIABLE_TYPE,CONSTANT_I,DEFAULT_VALUE,
    RUNTIME_EXEC_I )
  VALUES ( UPPER('SAD.bas_dml_lookup_pkg'), 'B', UPPER('g_pkg_name'),
           UPPER('VARCHAR2(30)'), TRUE, 'bas_dml_ic_price_rule_pkg',
           FALSE );

  INSERT INTO MIG_ORA_EXT.MIG_PKG_VARIABLES
  ( PACKAGE_NAME,SPEC_OR_BODY,VARIABLE_NAME,
    VARIABLE_TYPE,CONSTANT_I,DEFAULT_VALUE,
    RUNTIME_EXEC_I )
  VALUES ( UPPER('SAD.bas_dml_lookup_pkg'), 'B', UPPER('g_func_name'),
           UPPER('VARCHAR2(100)'), TRUE, 'bas_dml_ic_price_rule_pkg',
           FALSE );
END;
/
CREATE OR REPLACE FUNCTION SAD.bas_dml_lookup_pkg#bas_dml_lookup_pkg#func_name
  RETURN VARCHAR2
PACKAGE
  IS
    MIG_PV_VAL_DUMMY_G_PKG_NAME VARCHAR2(30) :=
      MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE('SAD', 'bas_dml_lookup_pkg', 'g_pkg_name')::VARCHAR2(30);
    MIG_PV_VAL_DUMMY_G_FUNC_NAME VARCHAR2(100) :=
      MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE('SAD', 'bas_dml_lookup_pkg', 'g_func_name')::VARCHAR2(100);
    l_func_name VARCHAR2 ( 100 )
    ...
```

---

### Data Warehouse Service

Tool Guide

6 Migration Tool - SQL Syntax Migration Tool

Issue 03 (2019-08-08) · Copyright © Huawei Technologies Co., Ltd.
BEGIN
    l_func_name := MIG_PV_VAL_DUMMY_G_PKG_NAME || '.' || MIG_PV_VAL_DUMMY_G_FUNC_NAME ;
    MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE
    ( 'SAD' ,'bas_dml_lookup_pkg' ,'g_func_name' ,MIG_PV_VAL_DUMMY_G_FUNC_NAME ) ;
    MIG_ORA_EXT.MIG_FN_SET_PKG_VARIABLE
    ( 'SAD' ,'bas_dml_lookup_pkg' ,'g_pkg_name' ,MIG_PV_VAL_DUMMY_G_PKG_NAME ) ;
    RETURN l_func_name ;
END ;
/
CREATE OR REPLACE PROCEDURE SAD.bas_dml_lookup_pkg#data_change_logs
    ( pi_table_name IN VARCHAR2
    , pi_table_key_columns IN VARCHAR2
    , po_error_msg OUT VARCHAR2 )
PACKAGE
IS
BEGIN

END ;
/
GRANT EXECUTE ON FUNCTION SAD.bas_dml_lookup_pkg#bas_dml_lookup_pkg#func_name() TO eip_sad;
GRANT EXECUTE ON FUNCTION SAD.bas_dml_lookup_pkg#data_change_logs(VARCHAR2, VARCHAR2) TO eip_sad;

6.13.11.9 Package Name List

Enable & Disable

Set

package_name_list = 'bas_lookup_misc_pkg';

Enable and disable are based on configuration parameter.

Input

If this parameter is enabled, the below line should be added before creating package objects.
SET package_name_list = '<<package name>>';
If it is not enabled, this line should not be added

Output

If this parameter is enabled, the below line should be added before creating package objects.
SET package_name_list = '<<package name>>';
If it is not enabled, this line should not be added.

6.13.11.10 Data Type

Subtype

The customized type in the package cannot be converted.

SUBTYPE error_msg IS sad_products_t.exception_description%TYPE;
SUBTYPE AR_FLAG IS SAD_RA_LINES_TI.AR_FLAG%TYPE;
SUBTYPE LOCK_FLAG IS SAD_SHIPMENT_BATCHES_T.LOCK_FLAG%TYPE;

Input
CREATE OR REPLACE PACKAGE SAD.bas_subtype_pkg IS
  SUBTYPE func_name IS sad_products_t.func_name%TYPE;
END bas_subtype_pkg;
/
CREATE OR REPLACE PACKAGE BODY SAD.bas_subtype_pkg IS
BEGIN
  NULL;
END bas_subtype_pkg;
/
CREATE OR REPLACE PACKAGE BODY SAD.bas_dml_lookup_pkg IS
  g_pkg_name CONSTANT VARCHAR2(30) := 'bas_dml_ic_price_rule_pkg';
  g_func_name VARCHAR2(100);
  FUNCTION func_name
  RETURN VARCHAR2
  IS
    l_func_name bas_subtype_pkg.func_name;
  BEGIN
    l_func_name := g_pkg_name || '.' || g_func_name;
    RETURN l_func_name;
  END func_name;
END bas_dml_lookup_pkg;
/
%ROWTYPE
Package Procedure/Function contains %ROWTYPE attribute in IN/OUT parameter and this is not supported
SCRIPTS : BAS_DML_SERVIECE_PKG.sql, BAS_LOOKUP_MISC_PKG.sql
INPUT :
CREATE OR REPLACE PACKAGE BODY "SAD"."BAS_DML_SERVIECE_PKG" IS
PROCEDURE save_split_ou(pi_split_ou IN split_ou%ROWTYPE,
  po_error_msg OUT VARCHAR2) IS
  ---
  BEGIN
  ---
  end save_split_ou;
end BAS_DML_SERVIECE_PKG;
OUTPUT :
CREATE OR REPLACE PROCEDURE SAD.BAS_DML_SERVIECE_PKG#save_split_ou ( pi_split_ou IN split_ou%ROWTYPE
  ,po_error_msg OUT VARCHAR2 ) IS MIG_PV_VAL_DUMMY_G_FUNC_NAME
  VARCHAR2 ( 30 ) := MIG_ORA_EXT.MIG_FN_GET_PKG_VARIABLE ( current_schema ( )
  ,'BAS_DML_SERVIECE_PKG'
  ,'g_func_name' ) ::VARCHAR2 ( 30 );
CREATE OR REPLACE PACKAGE BODY SAD.BAS_DML_SERVIECE_PKG IS
PROCEDURE save_split_ou(pi_split_ou IN split_ou%ROWTYPE,
po_error_msg OUT VARCHAR2) IS
BEGIN
UPDATE split_ou so
SET so.auto_balance_flag = pi_split_ou.auto_balance_flag,
    so.balance_start_date = pi_split_ou.balance_start_date,
    so.balance_source = pi_split_ou.balance_source
WHERE so.dept_code = pi_split_ou.dept_code;
EXCEPTION
WHEN OTHERS THEN
    po_error_msg := 'Others Exception raise in ' || g_func_name || ',' || SQLERRM;
END save_split_ou;
END bas_dml_serviece_pkg;
/

CREATE TYPE mig_typ_split_ou AS ...

CREATE OR REPLACE PROCEDURE SAD.BAS_DML_SERVIECE_PKG#save_split_ou
( pi_split_ou IN mig_typ_split_ou 
,po_error_msg OUT VARCHAR2 ) 
PACKAGE 
IS 
BEGIN 
UPDATE split_ou so 
SET so.auto_balance_flag = pi_split_ou.auto_balance_flag 
,so.balance_start_date = pi_split_ou.balance_start_date 
,so.balance_source = pi_split_ou.balance_source 
WHERE so.dept_code = pi_split_ou.dept_code ; 
EXCEPTION 
WHEN OTHERS THEN 
    po_error_msg := 'Others Exception raise in ' || g_func_name || ',' || SQLERRM ;
END ; 
/

6.13.12 Chinese Character Support

Refer 6.3.8 Chinese Character Support section for more information.

6.14 Oracle (Beta) Syntax Migration
6.14.1 Overview

The Oracle (Beta) Migration Syntax lists the Oracle features supported by the new (Beta) Syntax Migration tool, and for each feature provides the Oracle syntax and the equivalent GaussDB 100 syntax. The syntax listed in this section illustrates the internal migration logic used to migrate the Oracle scripts.

It is also a reference for the database migration team and for customer site verification of Oracle scripts migration.

6.14.2 Schema Objects

6.14.2.1 Overview

CREATE TABLE

The Oracle CREATE TABLE statements are used to create new tables. The target database supports Oracle CREATE TABLES without any migration.

ALTER TABLE

The Oracle ALTER TABLE statement is used to add, rename, modify, or drop/delete columns in a table. The target database supports Oracle ALTER TABLES without any migration.

6.14.2.2 Tables

PARTITIONS

Maintenance of large tables and indexes can become very time and resource consuming. At the same time, data access performance can reduce drastically for these objects. Partitioning of tables and indexes can benefit the performance and maintenance in several ways.

Figure 6-60 Partitioning and Sub-partitioning of Tables

Migration Tool supports migration of Range partition.

The tool does not support the following partitions/subpartitions and these are commented in the migrated scripts:

- List partition
- **Hash partition**
- **Range Subpartition**
- **List Subpartition**
- **Hash Subpartition**

The unsupported partitions/subpartitions may be supported in the future. Configuration parameters have been provided to enable/disable commenting of the unsupported statements. Refer [Configuration Parameters for Oracle Features](#).

- **PARTITION by HASH**
  Hash partitioning is a partitioning technique where a hash algorithm is used to distribute rows evenly across the different partitions (sub-tables). This is typically used where ranges are not appropriate, for example employee ID, product ID, and so on. Migration Tool does not support PARTITION and SUBPARTITION by HASH and will comment these statements.

**Input - HASH PARTITION**

```sql
CREATE TABLE sales
( prod_id NUMBER(6),
  cust_id NUMBER,
  time_id DATE,
  channel_id CHAR(1),
  promo_id NUMBER(6),
  quantity_sold NUMBER(3),
  amount_sold NUMBER(10,2)
)
PARTITION BY RANGE (time_id) SUBPARTITION BY HASH (cust_id)
SUBPARTITIONS 8 STORE IN (ts1, ts2, ts3, ts4)
( PARTITION sales_q1_2006 VALUES LESS THAN (TO_DATE('01-APR-2006','dd-MON-yyyy')) storage
  (initial 10T),
  PARTITION sales_q2_2006 VALUES LESS THAN (TO_DATE('01-JUL-2006','dd-MON-yyyy'))
  , PARTITION sales_q3_2006 VALUES LESS THAN (TO_DATE('01-OCT-2006','dd-MON-yyyy'))
  , PARTITION sales_q4_2006 VALUES LESS THAN (TO_DATE('01-JAN-2007','dd-MON-yyyy')) pctused 10
);```

**Output**

```sql
CREATE TABLE sales
( prod_id NUMBER(6),
  cust_id NUMBER,
  time_id DATE,
  channel_id CHAR(1),
  promo_id NUMBER(6),
  quantity_sold NUMBER(3),
  amount_sold NUMBER(10,2)
)
PARTITION BY RANGE (time_id) /*SUBPARTITION BY HASH (cust_id)
SUBPARTITIONS 8 STORE IN (ts1, ts2, ts3, ts4)*/
( PARTITION sales_q1_2006 VALUES LESS THAN (TO_DATE('01-APR-2006','dd-MON-yyyy'))
  , PARTITION sales_q2_2006 VALUES LESS THAN (TO_DATE('01-JUL-2006','dd-MON-yyyy'))
  , PARTITION sales_q3_2006 VALUES LESS THAN (TO_DATE('01-OCT-2006','dd-MON-yyyy'))
  , PARTITION sales_q4_2006 VALUES LESS THAN (TO_DATE('01-JAN-2007','dd-MON-yyyy'))
);```

- **PARTITION by LIST**
  List partitioning is a partitioning technique where you specify a list of discrete values for the partitioning key in the description for each partition. Migration Tool does not support PARTITION and SUBPARTITION by LIST and will comment these statements.

**Input - LIST PARTITION**

```sql
CREATE TABLE q1_sales_by_region
( deptno NUMBER

```
CREATE TABLE q1_sales_by_region
  ( deptno NUMBER,
    deptname varchar2 (20),
    quarterly_sales NUMBER (10,2),
    state varchar2 (2)
  )
PARTITION BY LIST (state)
  ( PARTITION q1_northwest
    VALUES ( 'OR', 'WA' ),
    PARTITION q1_southwest
    VALUES ( 'AZ', 'UT', 'NM' ),
    PARTITION q1_northeast
    VALUES ( 'NY', 'VM', 'NJ' ),
    PARTITION q1_southcentral
    VALUES ( 'OK', 'TX' )
  );

Output

CREATE TABLE q1_sales_by_region
  ( deptno NUMBER,
    deptname varchar2 (20),
    quarterly_sales NUMBER (10,2),
    state varchar2 (2)
  )
/*PARTITION BY LIST (state)
  ( PARTITION q1_northwest
    VALUES ( 'OR', 'WA' ),
    PARTITION q1_southwest
    VALUES ( 'AZ', 'UT', 'NM' ),
    PARTITION q1_northeast
    VALUES ( 'NY', 'VM', 'NJ' ),
    PARTITION q1_southcentral
    VALUES ( 'OK', 'TX' )
  )*/;

Input - LIST PARTITION with SUBPARTITIONS

CREATE TABLE q1_sales_by_region
  ( deptno NUMBER,
    deptname varchar2 (20),
    quarterly_sales NUMBER (10,2),
    state varchar2 (2)
  )
PARTITION BY LIST (state)
  SUBPARTITION BY RANGE (quarterly_sales)
  SUBPARTITION TEMPLATE ( SUBPARTITION original VALUES LESS THAN (1001),
    SUBPARTITION acquired VALUES LESS THAN (8001),
    SUBPARTITION recent VALUES LESS THAN (MAXVALUE) )
  ( PARTITION q1_northwest VALUES ( 'OR', 'WA' ),
    PARTITION q1_southwest VALUES ( 'AZ', 'UT', 'NM' ),
    PARTITION q1_northeast VALUES ( 'NY', 'VM', 'NJ' ),
    PARTITION q1_southcentral VALUES ( 'OK', 'TX' )
  );

Output

CREATE TABLE q1_sales_by_region
  ( deptno NUMBER,
    deptname varchar2 (20),
    quarterly_sales NUMBER (10,2),
    state varchar2 (2)
  )
PARTITION BY LIST (state)
  ( PARTITION q1_northwest
    VALUES ( 'OR', 'WA' ),
    PARTITION q1_southwest
    VALUES ( 'AZ', 'UT', 'NM' ),
    PARTITION q1_northeast
    VALUES ( 'NY', 'VM', 'NJ' ),
    PARTITION q1_southcentral
    VALUES ( 'OK', 'TX' )
  );
PARTITION BY RANGE

Range partitioning is a partitioning technique where ranges of data is stored separately in different sub-tables. Range partitioning is useful when you have distinct ranges of data you want to store together, for example the date field. Migration Tool supports PARTITION by RANGE. It does not support SUBPARTITION for RANGE and will comment these statements.

**Input - RANGE PARTITION (With Storage Parameters)**

```sql
CREATE TABLE CCM_TA550002_H (  STRU_ID VARCHAR2 (10) , ORGAN1_NO VARCHAR2 (10) , ORGAN2_NO VARCHAR2 (10) ) partition by range (ORGAN2_NO) (  partition CCM_TA550002_01 VALUES LESS than ('00100') /*TABLESPACE users*/ ,/*pctfree 10*/ ,/*initrans 1*/ ,/*storage(initial 256K NEXT 256K minextents 1 maxextents unlimited )*/ ,.partition CCM_TA550002_02 VALUES LESS than ('00200') /*TABLESPACE users*/ ,/*pctfree 10*/ ,/*initrans 1*/ ,/*storage(initial 256K NEXT 256K minextents 1 maxextents unlimited )*/ ,pctincrease 0 )*/ ;
```

**Output**

```sql
CREATE TABLE CCM_TA550002_H (  STRU_ID VARCHAR2 (10) , ORGAN1_NO VARCHAR2 (10) , ORGAN2_NO VARCHAR2 (10) ) partition by range (ORGAN2_NO) (  partition CCM_TA550002_01 VALUES LESS than ('00100') /*TABLESPACE users*/ ,/*pctfree 10*/ ,/*initrans 1*/ ,/*storage(initial 256K NEXT 256K minextents 1 maxextents unlimited )*/ ,.partition CCM_TA550002_02 VALUES LESS than ('00200') /*TABLESPACE users*/ ,/*pctfree 10*/ ,/*initrans 1*/ ,/*storage(initial 256K NEXT 256K minextents 1 maxextents unlimited )*/ ,pctincrease 0 )*/ ;
```
### Input - RANGE PARTITION with SUBPARTITIONS

```
CREATE TABLE composite_rng_list (
cust_id NUMBER(10),
cust_name VARCHAR2(25),
cust_state VARCHAR2(2),
time_id DATE)
PARTITION BY RANGE(time_id)
SUBPARTITION BY LIST (cust_state)
SUBPARTITION TEMPLATE(
    SUBPARTITION west VALUES ('OR', 'WA') TABLESPACE part1,
    SUBPARTITION east VALUES ('NY', 'CT') TABLESPACE part2,
    SUBPARTITION cent VALUES ('OK', 'TX') TABLESPACE part3)
    (PARTITION per1 VALUES LESS THAN (TO_DATE('01/01/2000','DD/MM/YYYY')),
     PARTITION per2 VALUES LESS THAN (TO_DATE('01/01/2005','DD/MM/YYYY')),
     PARTITION per3 VALUES LESS THAN (TO_DATE('01/01/2010','DD/MM/YYYY')),
     PARTITION future VALUES LESS THAN(MAXVALUE));
```

### Output

```
CREATE TABLE composite_rng_list (
cust_id NUMBER(10),
cust_name VARCHAR2(25),
cust_state VARCHAR2(2),
time_id DATE)
PARTITION BY RANGE(time_id)
/*SUBPARTITION BY LIST (cust_state)
SUBPARTITION TEMPLATE(
    SUBPARTITION west VALUES ('OR', 'WA') TABLESPACE part1,
    SUBPARTITION east VALUES ('NY', 'CT') TABLESPACE part2,
    SUBPARTITION cent VALUES ('OK', 'TX') TABLESPACE part3)*/
    (PARTITION per1 VALUES LESS THAN (TO_DATE('01/01/2000','DD/MM/YYYY')),
     PARTITION per2 VALUES LESS THAN (TO_DATE('01/01/2005','DD/MM/YYYY')),
     PARTITION per3 VALUES LESS THAN (TO_DATE('01/01/2010','DD/MM/YYYY')),
     PARTITION future VALUES LESS THAN(MAXVALUE));
```

### Input - RANGE PARTITION with SUBPARTITION TEMPLATE

```
CREATE TABLE composite_rng_rng (
cust_id NUMBER(10),
cust_name VARCHAR2(25),
cust_state VARCHAR2(2),
time_id DATE)
PARTITION BY RANGE(time_id)
SUBPARTITION BY RANGE (cust_id)
SUBPARTITION TEMPLATE(
    SUBPARTITION original VALUES LESS THAN (1001) TABLESPACE part1,
    SUBPARTITION acquired VALUES LESS THAN (8001) TABLESPACE part2,
    SUBPARTITION recent VALUES LESS THAN (MAXVALUE) TABLESPACE part3)
    (PARTITION per1 VALUES LESS THAN (TO_DATE('01/01/2000','DD/MM/YYYY')),
     PARTITION per2 VALUES LESS THAN (TO_DATE('01/01/2005','DD/MM/YYYY')),
     PARTITION per3 VALUES LESS THAN (TO_DATE('01/01/2010','DD/MM/YYYY')),
     PARTITION future VALUES LESS THAN (MAXVALUE));
```

### Output

```
CREATE TABLE composite_rng_rng (
cust_id NUMBER(10),
cust_name VARCHAR2(25),
cust_state VARCHAR2(2),
time_id DATE)
PARTITION BY RANGE(time_id)
/*SUBPARTITION BY RANGE (cust_id)
SUBPARTITION TEMPLATE(
    SUBPARTITION original VALUES LESS THAN (1001) TABLESPACE part1,
    SUBPARTITION acquired VALUES LESS THAN (8001) TABLESPACE part2,
    SUBPARTITION recent VALUES LESS THAN (MAXVALUE) TABLESPACE part3)*/
    (PARTITION per1 VALUES LESS THAN (TO_DATE('01/01/2000','DD/MM/YYYY')),
     PARTITION per2 VALUES LESS THAN (TO_DATE('01/01/2005','DD/MM/YYYY')),
     PARTITION per3 VALUES LESS THAN (TO_DATE('01/01/2010','DD/MM/YYYY')),
     PARTITION future VALUES LESS THAN (MAXVALUE));
```
PCTINCREASE

The storage parameter PCTINCREASE is not supported for all the tables. In addition, all storage parameters (like pctfree, minextents, maxextents) are not allowed for Partition tables.

Input - TABLE with PCTINCREASE

```
CREATE TABLE divisions
    (div_no     NUMBER(2),
     div_name   VARCHAR2(14),
     location   VARCHAR2(13) )
    STORAGE  ( INITIAL 100K  NEXT     50K
              MINEXTENTS 1  MAXEXTENTS 50  PCTINCREASE 5);
```

Output

```
CREATE TABLE divisions
    (div_no     NUMBER(2),
     div_name   VARCHAR2(14),
     location   VARCHAR2(13) )
    STORAGE  ( INITIAL 100K  NEXT     50K
              MINEXTENTS 1  MAXEXTENTS 50    );
```

FOREIGN KEY

A foreign key is a way to enforce referential integrity within an Oracle database. A foreign key means that values in one table must also appear in another table. The referenced table is called the parent table while the table with the foreign key is called the child table. The foreign key in the child table will generally reference a primary key in the parent table. A foreign key can be defined in either a CREATE TABLE statement or an ALTER TABLE statement.

A foreign key **constraint** must be established with the **reference-clause**. An inline constraint clause appears as part of the column definition clause or the object properties clause. An out-of-line constraint appears as part within a relational properties clause or the object properties clause.

If the configuration parameter **foreignKeyHandler** is set to true (default value), then the tool will migrate these statements into commented statements.

Migration tool supports inline and out-of-line foreign key constraints, excluding REF constraints, as shown in the following examples.
NOTE

Migration tool does not support inline and out-of-line REF constraints on columns which do not specify data types. The unsupported statements will be copied verbatim.

- Example: Foreign Key with inline REF constraint
  ```sql
  CREATE TABLE dept_20 (
    employee_id NUMBER(4),
    salary NUMBER(7,2),
    department_id CONSTRAINT fk_deptno /* inline ref */
    REFERENCES departments(department_id)
  );
  ```

- Example: Foreign Key with out-of-line REF constraint
  ```sql
  CREATE TABLE dept_20 (
    employee_id NUMBER(4),
    salary NUMBER(7,2),
    department_id,
    CONSTRAINT fk_deptno /* out of line ref */
    FOREIGN KEY (department_id)
    REFERENCES departments(department_id)
  );
  ```

Input - Foreign Key with inline constraint in CREATE TABLE

```
CREATE TABLE orders (
  order_no INT NOT NULL PRIMARY KEY,
  order_date DATE NOT NULL,
  cust_id INT
  CONSTRAINT fk_orders_cust
  REFERENCES customers(cust_id)
  ON DELETE SET NULL
  INITIALLY DEFERRED
  ENABLE NOVALIDATE
);
```

Output

```
CREATE TABLE orders (
  order_no INT NOT NULL PRIMARY KEY,
  order_date DATE NOT NULL,
  cust_id INT
  CONSTRAINT fk_orders_cust
  REFERENCES customers(cust_id)
  ON DELETE SET NULL
  INITIALLY DEFERRED
  ENABLE NOVALIDATE
);
```

Input - Foreign Key with out-of-line constraint in CREATE TABLE

```
CREATE TABLE orders (
  order_no INT NOT NULL,
  order_date DATE NOT NULL,
  cust_id INT NOT NULL,
  PRIMARY KEY (order_no),
  CONSTRAINT fk_orders_cust
  FOREIGN KEY (cust_id)
  REFERENCES customers(cust_id)
  ON DELETE CASCADE);
```

Output

```
CREATE TABLE orders (
  order_no INT NOT NULL,
  order_date DATE NOT NULL,
  cust_id INT NOT NULL,
  PRIMARY KEY (order_no),
  CONSTRAINT fk_orders_cust
  REFERENCES customers(cust_id)
  ON DELETE CASCADE);
```

Data Warehouse Service
Tool Guide

6 Migration Tool - SQL Syntax Migration Tool

Copyright © Huawei Technologies Co., Ltd.
FOREIGN KEY (cust_id) 
REFERENCES customers(cust_id) 
ON DELETE CASCADE*/);

Input - Foreign Key with inline constraint in ALTER TABLE

ALTER TABLE orders ADD ( cust_id INT 
CONSTRAINT fk_orders_cust 
REFERENCES customers(cust_id) 
ON DELETE SET NULL 
INITIALLY DEFERRED 
ENABLE NOVALIDATE );

Output

ALTER TABLE orders ADD ( cust_id INT 
/*CONSTRAINT fk_orders_cust 
REFERENCES customers(cust_id) 
ON DELETE SET NULL 
INITIALLY DEFERRED 
ENABLE NOVALIDATE*/ );

Input - Foreign Key Add Constraint in ALTER TABLE

ALTER TABLE porders 
ADD CONSTRAINT fk_orders_cust 
FOREIGN KEY (cust_id) 
REFERENCES customers(cust_id);

Output

/*ALTER TABLE porders 
/*ADD /*CONSTRAINT fk_orders_cust 
FOREIGN KEY (cust_id) 
REFERENCES customers(cust_id)*/*/;/

LONG Data Type

Columns defined as LONG can store variable-length character data containing up to 2 gigabytes of information. The tool supports LONG data types in TABLE structure and PL/SQL.

See also: LONG data-type in PL/SQL

Input - LONG data-type in table structure

CREATE TABLE project ( proj_cd INT 
, proj_name VARCHAR2(32) 
, dept_no INT 
, proj_det LONG );

Output

CREATE TABLE project ( proj_cd INT 
, proj_name VARCHAR2(32) 
, dept_no INT 
, proj_det TEXT );

Table Whose Name is Same as Constraint Name

NOTE

If synonyms/primary key constraints are case insensitive, they will be replaced by lowercase version.

Input
CREATE TABLE public.rp_t_transactiondetail
(
policyno VARCHAR2(22) NOT NULL,
proposalno VARCHAR2(22),
businessnature VARCHAR2(9),
businessnaturename VARCHAR2(360),
clausetype VARCHAR2(9),
carchecker VARCHAR2(60),
carkindcode VARCHAR2(9),
usenaturecode VARCHAR2(9)
);

ALTER TABLE public.rp_t_transactiondetail ADD CONSTRAINT mig_rp_t_transactiondetail PRIMARY KEY (policyno);

/*ALTER TABLE public.rp_t_transactiondetail ADD CONSTRAINT rp_t_transactiondetail PRIMARY KEY (policyno);*/

6.14.2.3 Temporary Tables

Since DWS do not support schema for temporary tables. Special handling is required for such cases.

Double quotes ("”) are used in to specify case sensitiveness. The following cases are possible:

<table>
<thead>
<tr>
<th>Input Use Cases</th>
<th>Case Sensitivity</th>
<th>Migration Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>● schema.tablename</td>
<td>Both schema name and table name are case insensitive.</td>
<td>The temporary table name is replaced by schema_tablename.</td>
</tr>
<tr>
<td>● Schema.tableName</td>
<td></td>
<td></td>
</tr>
<tr>
<td>● SCHEMA.TABLENAMES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>● &quot;schema&quot;.tablename</td>
<td>Schema name is case sensitive, table name is not.</td>
<td>The temporary table name will be replaced by schema_tablename.</td>
</tr>
<tr>
<td>● &quot;schema&quot;.&quot;tablename&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>● All other cases</td>
<td>-</td>
<td>The temporary table name is NOT replaced.</td>
</tr>
</tbody>
</table>
### Input - Temporary table with On Commit Delete Rows

```
CREATE GLOBAL TEMPORARY TABLE schema1.temp_tbl1
  ( col1 VARCHAR2 (400)
    , col2 DATE NOT NULL
  )
ON COMMIT DELETE ROWS TABLESPACE test;
```

**Output**

```
CREATE LOCAL TEMPORARY TABLE schema1_temp_tbl1
  ( col1 VARCHAR2 (400)
    , col2 DATE NOT NULL
  )
ON COMMIT PRESERVE ROWS TABLESPACE test;
```

### 6.14.2.4 Indexes

An index is a performance-tuning method of allowing faster retrieval of records. An index creates an entry for each value that appears in the indexed columns. In DWS, during index creation, schema name cannot be specified along with index name. The index will be automatically created in the schema where the index table is created.

#### Input - Create Index Name with Schema Name

```
CREATE INDEX scott.ix_tab1_col1 ON scott.tab1 (col1)
  /*TABLESPACE users*/
  pctfree 10 initrans 2
  storage ( initial 256 K NEXT 256 K
  minextents 1 maxextents
  unlimited );
```

**Output**

```
CREATE INDEX ix_tab1_col1 ON scott.tab1 (col1)
  /*TABLESPACE users*/
  pctfree 10 initrans 2
  storage ( initial 256 K NEXT 256 K
  minextents 1 maxextents
  unlimited );
```

#### Input - Local Partition Index : Range Partition table with Global Partition index

```
CREATE TABLE sales
  ( prod_id NUMBER(6)
    , quantity_sold NUMBER(3)
    , amount_sold NUMBER(10,2)
    , time_id DATE
  )
PARTITION BY RANGE (time_id)
  ( PARTITION sales_q1_2006 VALUES LESS THAN (TO_DATE('01-APR-2006'
    , 'dd-MON-yyyy'))
    /*TABLESPACE users*/
    , PARTITION sales_q2_2006 VALUES LESS THAN (TO_DATE('01-JUL-2006'
    , 'dd-MON-yyyy'))
    /*TABLESPACE users*/
  )
;
CREATE INDEX amount_sold_ix ON sales(amount_sold, quantity_sold)
  GLOBAL PARTITION BY
  RANGE(amount_sold)
  ( PARTITION p_100 VALUES LESS THAN
    (100)
```
CREATE TABLE sales
( prod_id NUMBER(6),
  quantity_sold NUMBER(3),
  amount_sold NUMBER(10,2),
  time_id DATE )
PARTITION BY RANGE (time_id)
( PARTITION sales_q1_2006 VALUES LESS THAN (TO_DATE('01-APR-2006','dd-MON-yyyy'))
  /*TABLESPACE users*/,
  PARTITION sales_q2_2006 VALUES LESS THAN (TO_DATE('01-JUL-2006','dd-MON-yyyy'))
  /*TABLESPACE users*/);

CREATE INDEX amount_sold_ix ON sales(amount_sold, quantity_sold)
LOCAL;

CREATE TABLE sales_by_region
( deptno NUMBER,
  deptname varchar2 (20),
  quarterly_sales NUMBER (10,2),
  quantity_sold NUMBER(3),
  state varchar2 (2) )
PARTITION BY LIST (state)
( PARTITION q1_northwest VALUES ( 'OR', 'WA' )
  , PARTITION q1_southwest VALUES ( 'AZ', 'UT', 'NM' )
  , PARTITION q1_northeast VALUES ( 'NY', 'VM', 'NJ' )
  , PARTITION q1_southcentral VALUES ( 'OK', 'TX' ) );

CREATE INDEX sale_by_reg_ix ON
sales_by_region(quarterly_sales,quantity_sold)
GLOBAL PARTITION BY RANGE(quarterly_sales)
( PARTITION p_100 VALUES LESS THAN (100)
  , PARTITION p_1000 VALUES LESS THAN (1000)
  , PARTITION p_greater_than_1000000 VALUES LESS THAN (maxvalue) );

CREATE TABLE sales_by_region
( deptno NUMBER,
  deptname varchar2 (20),
  quarterly_sales NUMBER (10,2),
  quantity_sold NUMBER(3),
  state varchar2 (2) )
/*PARTITION BY LIST (state)
( PARTITION q1_northwest VALUES ( 'OR', 'WA' )
  , PARTITION q1_southwest VALUES ( 'AZ', 'UT', 'NM' )
  , PARTITION q1_northeast VALUES ( 'NY', 'VM', 'NJ' )
  , PARTITION q1_southcentral VALUES ( 'OK', 'TX' ) */;

CREATE INDEX sale_by_reg_ix ON
sales_by_region(quarterly_sales,quantity_sold)
/*GLOBAL PARTITION BY RANGE(quarterly_sales)
( PARTITION p_100 VALUES LESS THAN (100)
  , PARTITION p_1000 VALUES LESS THAN (1000)
  , PARTITION p_greater_than_1000000 VALUES LESS THAN (maxvalue) */;
Input - Local Partition Index: Range Partition table with Local Partition index

```
CREATE TABLE sales2
(
  prod_id NUMBER(6),
  quantity_sold NUMBER(3),
  amount_sold NUMBER(10,2),
  time_id DATE
)
PARTITION BY RANGE (time_id)
  (PARTITION sales2_q1_2006 VALUES LESS THAN (TO_DATE('01-APR-2006','dd-MON-yyyy')) /*TABLESPACE users*/,
   PARTITION sales2_q2_2006 VALUES LESS THAN (TO_DATE('01-JUL-2006','dd-MON-yyyy')) /*TABLESPACE users*/);

CREATE INDEX amount_sold_ix ON sales2(amount_sold)
  LOCAL
   (PARTITION sales_q1_ix,
    PARTITION sales_q2_ix);
```

Output

```
CREATE TABLE sales2
(
  prod_id NUMBER(6),
  quantity_sold NUMBER(3),
  amount_sold NUMBER(10,2),
  time_id DATE
)
PARTITION BY RANGE (time_id)
  (PARTITION sales2_q1_2006 VALUES LESS THAN (TO_DATE('01-APR-2006','dd-MON-yyyy')) /*TABLESPACE users*/,
   PARTITION sales2_q2_2006 VALUES LESS THAN (TO_DATE('01-JUL-2006','dd-MON-yyyy')) /*TABLESPACE users*/);

CREATE INDEX amount_sold_ix ON sales2(amount_sold)
  LOCAL
   (PARTITION sales_q1_ix,
    PARTITION sales_q2_ix);
```

Input - Local Partition Index: Range Partition table having Partition index without partition name

```
CREATE TABLE sales
(
  prod_id NUMBER(6),
  quantity_sold NUMBER(3),
  amount_sold NUMBER(10,2),
  time_id DATE
)
PARTITION BY RANGE (time_id)
  (PARTITION sales_q1_2006 VALUES LESS THAN (TO_DATE('01-APR-2006','dd-MON-yyyy'))/*TABLESPACE users*/,
   PARTITION sales_q2_2006 VALUES LESS THAN (TO_DATE('01-JUL-2006','dd-MON-yyyy'))/*TABLESPACE users*/);

CREATE INDEX amount_sold_idx ON sales(amount_sold) LOCAL;
```

Output

```
CREATE TABLE sales
(
  prod_id NUMBER(6),
  quantity_sold NUMBER(3),
  amount_sold NUMBER(10,2),
  time_id DATE
)
PARTITION BY RANGE (time_id)
  (PARTITION sales_q1_2006 VALUES LESS THAN (TO_DATE('01-APR-2006','dd-MON-yyyy'))/*TABLESPACE users*/,
   PARTITION sales_q2_2006 VALUES LESS THAN (TO_DATE('01-JUL-2006','dd-MON-yyyy'))/*TABLESPACE users*/);

CREATE INDEX amount_sold_idx ON sales(amount_sold) LOCAL;
```
CREATE INDEX WITH REVERSE KEYWORD

CREATE INDEX IDX_DB2_CONVER_DTL ON SEAS_DB2_CONVER_DTL (BEATCH_ID) REVERSE
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE DRMS ;

6.14.2.5 Sequences

SEQUENCE is an Oracle object used to create a number sequence. They are used to handle autonumber fields and to create unique numbers to act as primary keys.

Use the maxValInSequence parameter to set the maximum value supported in a sequence.

**NOTE**

If the max value of a sequence is greater than the maximum value configured for maxValInSequence, then the configured max value will replace the input max value in the migrated script.

SEQUENCE

**Input - SEQUENCE**

CREATE SEQUENCE GROUP_DEF_SEQUENCE
  minvalue 1
  maxvalue 10000000000000000000
  start with 1152
  increment by 1
  cache 50;

**Output** (maxValInSequence =1000000)
CREATE SEQUENCE GROUP_DEF_SEQUENCE
minvalue 1
maxvalue 1000000
start with 1152
increment by 1
cache 50;

**Input - SEQUENCE with ORDER**

CREATE SEQUENCE GROUP_DEF_SEQUENCE
minvalue 1 maxvalue 1000 start with 11 increment by 1 cache 50 order;

**Output**

CREATE SEQUENCE GROUP_DEF_SEQUENCE
minvalue 1 maxvalue 1000 start with 11 increment by 1 cache 50 /*order*/;

**Input - SEQUENCE with NOCACHE**

CREATE SEQUENCE GROUP_DEF_SEQUENCE
minvalue 1 maxvalue 9223372036854775807 start with 11 increment by 1 nocache;

**Output**

CREATE SEQUENCE GROUP_DEF_SEQUENCE
minvalue 1 maxvalue 9223372036854775807 start with 11 increment by 1 /*nocache*/;

**ALTER SEQUENCE**

**ALTER SEQUENCE** is used in Oracle to change the increment, minimum and maximum values, cached numbers, and behavior of a sequence. The tool will comment the **ALTER SEQUENCE** statements in the migrated scripts.

**Input - ALTER SEQUENCE**

ALTER SEQUENCE customers_seq MAXVALUE 1500;

**Output**

/*ALTER SEQUENCE customers_seq MAXVALUE 1500;*/

**NEXTVAL**

**NEXTVAL** is an Oracle system function and is not implicitly supported in DWS. To support this function, the Migration tool creates a **NEXTVAL** function in the PUBLIC schema. The migrated statements will use this new function **PUBLIC.NEXTVAL**.

**Input - SEQUENCE with NEXTVAL**

"SELECT 'X' || TO_CHAR (SYSDATE,'yyyyymmdd')
|| LPAD (SEQ_SPMS_SYSSOFT_ASSETS_ID.nextval, '6', '0')
INTO O_SOFTASSETS_ID
FROM DUAL;"

**Output**

"SELECT 'X' || TO_CHAR( SYSDATE ,'yyyyymmdd' )
|| LPAD( nextval ( 'SEQ_SPMS_SYSSOFT_ASSETS_ID' ), '6', '0' )
INTO O_SOFTASSETS_ID
FROM DUAL ;"
CURRVAL

CURRVAL is an Oracle system function and is not implicitly supported in DWS. To support this function, the Migration tool creates a CURRVAL function in the PUBLIC schema. The migrated statements will use this new function PUBLIC.CURRVAL.

Input - SEQUENCE with CURRVAL

CREATE SEQUENCE temp_seq minvalue 1 maxvalue 100 start WITH 11 increment BY 1 cache 50;
CREATE TABLE temp
    (id INTEGER PRIMARY KEY,
    value INTEGER DEFAULT temp_seq.currval);

Output

CREATE SEQUENCE temp_seq minvalue 1 maxvalue 100 start WITH 11 increment BY 1 cache 50;
CREATE TABLE temp
    (id INTEGER PRIMARY KEY,
    value INTEGER DEFAULT CURRVAL('temp_seq'));

6.14.2.6 Purge

In Oracle, DROP TABLE statement moves a table to the recycle bin. The PURGE statement is used to remove a table or index from recycle bin and release all of the space associated with the object. This statement is also used to remove the entire recycle bin. The migrated query also does not contain PURGE.

Input - PURGE

PURGE TABLESPACE testts USER test;

Output

/*PURGE TABLESPACE testts USER test;*/

6.14.2.7 Database Keywords

Database keyword LIMIT is supported by the Migration Tool by enclosing the migrated keyword in double-quotes (").

Input - Database Keyword LIMIT

create table test3 (limit number, col2 number, constraint check_limit_valid check (limit >= 0));

Output

create table test3 ("LIMIT" number, col2 number, constraint check_limit_valid check ("LIMIT" >= 0));

6.14.2.8 Synonym

In Oracle, synonym is an alternative name for objects such as tables, views, sequences, stored procedures, and other database objects.
If synonyms/primary key constraints are case insensitive, they will be replaced by lowercase version.

Synonyms for table are supported in oracle beta. Synonyms will be replaced in SELECT statement by the corresponding table names.

Input

create or replace synonym PRP_C_INSURED for KHYX_EA.PRP_C_INSURED;
sel ect PRP_C_INSURED.proposal_no_,
       PRP_C_INSURED.comid_,
       PRP_C_INSURED.insured_name_
from prp_c_INSURED
where PRP_C_INSURED.insured_flag_ = '100000000000000000000000000000'
or PRP_C_INSURED.insured_flag_ = '110000000000000000000000000000';

Output

/*create or replace synonym PRP_C_INSURED for KHYX_EA.PRP_C_INSURED;*/
sel ect PRP_C_INSURED.proposal_no_,
       PRP_C_INSURED.comid_,
       PRP_C_INSURED.insured_name_
from khyx_ea.prp_c_insured prp_c_INSURED
where PRP_C_INSURED.insured_flag_ = '100000000000000000000000000000'
or PRP_C_INSURED.insured_flag_ = '110000000000000000000000000000';

6.14.2.9 Gauss Keyword

Column alias keyword

owner,key,begin,label,version,text,year,position,function,end,procedure,limit,spl it,na me,value,
app,action,role,comments,fileheader,language,header,type,content,show,maxvalue,
offset,return,time,authorization

These keywords are reserved words for the Gauss database."AS" should be added when these keywords using as a column alias in SQL Query.

Input

select col1 owner from tab1;

Output

select col1 as owner from tab1;

Database object keyword

The following keywords are reserved words for the Gauss database:

- function
- end
- procedure
- limit
- split
- maxvalue
- offset
- return
Double quotes should be added for these keywords when used as a database object name in an input script. The database objects in context are referred to table, column, view, function, and stored procedure.

**Input**

```sql
select limit AS col1 from tab1;
select col1 from limit;
cREATE view limit;
cREATE function limit...
cREATE procedure limit...;
```

**Output**

```sql
select "limit" AS col1 from tab1;
select col1 from "limit";
cREATE view "limit"...
cREATE function "limit"...
cREATE procedure "limit"...;
```

### 6.14.3 DML

#### 6.14.3.1 Overview

This section contains the migration syntax for migrating Oracle DML. The migration syntax decides how the supported keywords/features are migrated.

#### 6.14.3.2 INSERT

**Overview**

The Oracle INSERT statement is used to insert a single record or multiple records into a table.

**NOTE**

For INSERT FIRST, comments present in the non SELECT clauses in the input are appended at the end of the output.

**INSERT ALL**

The Oracle INSERT ALL statement is used to add multiple rows using a single INSERT statement. The rows can be inserted into either a single table or multiple tables. The target query is converted as a common table expression (CTE).

**Input - INSERT ALL**

```sql
INSERT ALL
INTO ap_cust
VALUES ( customer_id, program_id, delivered_date )
INTO ap_orders ( ord_dt, Prg_id )
VALUES ( order_date, program_id )
SELECT program_id, delivered_date, customer_id, order_date
FROM order
WHERE deptno = 10;
```

**Output**

```sql
```
WITH Sel AS
( SELECT program_id, delivered_date, customer_id, order_date
  FROM order
  WHERE deptno = 10 )
, ins1 AS ( INSERT INTO ap_cust ( SELECT customer_id,
  program_id, delivered_date FROM Sel ) RETURNING 1 )
INSERT INTO ap_orders ( ord_dt, Prg_id ) ( SELECT order_date, program_id FROM Sel );

Input - INSERT ALL with condition
INSERT ALL
WHEN deptno <= 10 THEN
  INTO emp12
  VALUES ( empno, ename, job, mgr, hiredate, sal )
  WHEN deptno > 10 and deptno <= 20
  INTO emp13
  VALUES ( empno, ename, job, mgr, hiredate, sal )
WHEN deptno > 20 THEN
  INTO emp14
  VALUES ( empno, ename, job, mgr, hiredate, sal )
SELECT empno, ename, job, mgr,
  hiredate, sal, Deptno
FROM emp
  WHERE job='MANAGER';

Output
WITH Sel AS
( SELECT empno, ename, job, mgr, hiredate, sal, Deptno
  FROM emp
  WHERE job='MANAGER' )
, ins1 AS ( INSERT INTO emp12 ( SELECT empno, ename, job, mgr, hiredate, sal FROM Sel WHERE deptno <= 10 ) RETURNING 1 )
, ins2 AS ( INSERT INTO emp13 ( SELECT empno, ename, job, mgr, hiredate, sal FROM Sel WHERE deptno > 10 and deptno <= 20 ) RETURNING 1 )
INSERT INTO emp14 ( SELECT empno, ename, job, mgr, hiredate, sal FROM Sel WHERE deptno > 20 );

Input - INSERT ALL with ELSE
INSERT ALL
WHEN deptno <= 10 THEN
  INTO emp12
WHEN deptno > 10 AND deptno <= 20 THEN
  INTO emp13
ELSE
  INTO emp14 ( empno, ename, job )
    VALUES ( empno, ename, job )
SELECT empno, ename, job, mgr, hiredate, sal, deptno
FROM emp
WHERE job='MANAGER';

Output
WITH Sel AS
( SELECT empno, ename, job, mgr, hiredate, sal, deptno
  FROM emp
  WHERE job='MANAGER' )
, ins1 AS ( INSERT INTO emp12 ( SELECT empno, ename, job, mgr, hiredate, sal, deptno FROM Sel WHERE deptno <= 10 ) RETURNING 1 )
, ins2 AS ( INSERT INTO emp13 ( SELECT empno, ename, job, mgr, hiredate, sal, deptno FROM Sel WHERE deptno > 10 AND deptno <= 20 ) RETURNING 1 )
INSERT INTO emp14 ( empno, ename, job ) ( SELECT empno, ename, job FROM Sel )
MINUS SELECT empno, ename, job FROM Sel WHERE ( deptno <= 10 ) OR ( deptno > 10 AND deptno <= 20 );
**INSERT FIRST**

The Oracle `INSERT FIRST` is used to execute an insert statement associated with the first condition that is true; other statements are ignored. The target query is converted as a common table expression (CTE).

**Input - INSERT FIRST**

```sql
INSERT FIRST
WHEN deptno <= 10 THEN
    INTO emp12
    SELECT empno, ename, job, mgr, hiredate, sal, comm, deptno
FROM emp
WHERE deptno IS NOT NULL;
```

**Output**

```sql
WITH Sel AS
    ( SELECT ROW_NUMBER() OVER() Ins_First_RN,
        empno, ename, job, mgr, hiredate, sal, comm, deptno
        FROM emp
        WHERE deptno IS NOT NULL )
    , ins1 AS ( INSERT INTO emp12 (SELECT empno, ename, job, mgr, hiredate, sal, comm, deptno FROM Sel
        WHERE deptno <= 10 ) RETURNING 1)
INSERT INTO emp13 (SELECT empno, ename, job, mgr, hiredate, sal, comm, deptno FROM (SELECT * FROM Sel WHERE comm > 500 ) s1 LEFT JOIN (SELECT Ins_First_RN FROM Sel WHERE deptno <= 10 ) s2
    ON s1.Inst_First_RN = s2.Inst_First_RN WHERE s2.Inst_First_RN IS NULL )
```

**Input - INSERT FIRST with ELSE**

```sql
INSERT FIRST
WHEN ottl < 100000 THEN
    INTO small_orders
    VALUES ( oid, ottl, sid, cid,cl,cem )
WHEN ottl > 100000 AND ottl < 200000 THEN
    INTO medium_orders (moid, mottl, msid, mcid,mcl,mcem)
    VALUES ( oid, ottl, sid, cid,cl,cem )
ELSE
    INTO special_orders
    SELECT o.order_id oid, o.orders_total
    , o.sales_rep_id sid,
    c.credit_limit cl, c.cust_email cem
    FROM orders1 o, customers1 c
    WHERE c.customer_id = o.customer_id;
```

**Output**

```sql
WITH Sel AS
    ( SELECT ROW_NUMBER() OVER() Ins_First_RN,
        o.order_id oid, o.orders_total
        ottl, o.customer_id cid
        , o.sales_rep_id sid,
        c.credit_limit cl, c.cust_email cem
        FROM orders1 o, customers1 c
        WHERE c.customer_id = o.customer_id )
    , ins1 AS ( INSERT INTO small_orders (SELECT oid, ottl, sid,
        cid,cl,cem FROM Sel WHERE ottl < 100000 ) RETURNING 1 )
    , ins2 AS ( INSERT INTO medium_orders (moid, mottl, msid, mcid,mcl,mcem)
        (SELECT * FROM Sel WHERE ottl > 100000 AND ottl < 200000 ) s2
        ON s1.Inst_First_RN = s2.Inst_First_RN WHERE s2.Inst_First_RN IS NULL )
( (SELECT * FROM Sel WHERE ottl > 100000 AND ottl < 200000 ) s1 LEFT JOIN (SELECT Ins_First_RN FROM Sel WHERE ottl <= 10 ) s2
    ON s1.Inst_First_RN = s2.Inst_First_RN WHERE s2.Inst_First_RN IS NULL ) RETURNING 1 )
```
MERGE

MERGE with UPDATE and INSERT

(By default, the config parameter "mergeImplementation" is set to "WITH")

MERGE is an ANSI-standard SQL syntax operator used to select rows from one or more sources for updating or inserting into a table or view. The conditions to update or insert to the target table or view can be specified.

Currently, this feature is not supported by DWS. The Migration Tool uses multiple approaches to migrate MERGE into DWS compatible SQL.

The configuration parameter is mergeImplementation.

Input

MERGE INTO employees01 e
USING (SELECT empid, ename, startdate, address
FROM hr_records
WHERE empid > 100) h
ON (e.id = h.empid)
WHEN MATCHED THEN
UPDATESET e.address = h.address
, e.ename = h.ename
WHEN NOT MATCHED THEN
INSERT (empid,ename,startdate,address)
VALUES (h.empid,h.ename,h.startdate,h.address);

Output

WITH h AS ( SELECT empid, ename, startdate, address
FROM hr_records
WHERE empid > 100
)
, UPD_REC AS ( UPDATE employees01 e
SET e.address = h.address
, e.ename = h.ename
FROM h
WHERE e.id = h.empid
returning e.*
)
INSERT INTO employees01 (empid,ename,startdate,address)
SELECT h.empid, h.ename, h.startdate, h.address
FROM h
WHERE NOT EXISTS ( SELECT 1 FROM UPD_REC e
WHERE e.id = h.empid );

MERGE with UPDATE and INSERT

(If the config parameter "mergeImplementation" is set to "SPLIT")

Input

MERGE INTO employees01 e
USING (SELECT empid, ename, startdate, address
FROM hr_records
WHERE empid > 100) h
ON (e.id = h.empid)
WHEN MATCHED THEN
UPDATESET e.address = h.address
, e.ename = h.ename
WHEN NOT MATCHED THEN
INSERT (empid, ename, startdate, address)
VALUES (h.empid, h.ename, h.startdate, h.address);

Output

UPDATE employees01 e
SET e.address = h.address
  , e.ename = h.ename
FROM (SELECT empid, ename, startdate, address
    FROM hr_records
    WHERE empid > 100
  ) h
WHERE e.id = h.empid;
INSERT INTO employees01 (empid, ename, startdate, address)
SELECT h.empid, h.ename, h.startdate, h.address
FROM hr_records
WHERE empid > 100
) hLEFT OUTER JOIN employees01 e
ON e.id = h.empid
WHERE e.id IS NULL;

MERGE with only UPDATE

(By default, the config parameter "mergImplemented" is set to "WITH")

Input

MERGE INTO student a
USING (SELECT id, sname, score
FROM student_n) b
ON (a.id = b.id)
WHEN MATCHED THEN
UPDATE SET a.sname = b.sname
  , a.score = b.score
WHERE a.score > 600;

Output

WITH b AS (SELECT id, sname, score
FROM student_n )
UPDATE student a
SET a.sname = b.sname
  , a.score = b.score
FROM b
WHERE a.score > 600
AND a.id = b.id;

MERGE with only UPDATE

(If the config parameter "mergImplemented" is set to "SPLIT")

Input

MERGE INTO student a
USING (SELECT id, sname, score
FROM student_n) b
ON (a.id = b.id)
WHEN MATCHED THEN
UPDATE SET a.sname = b.sname
  , a.score = b.score
WHERE a.score > 600;

Output

UPDATE student a
SET a.sname = b.sname
  , a.score = b.score
FROM (SELECT id, sname, score
    FROM student_n
    WHERE score > 600
  ) b
WHERE a.id = b.id;
FROM student_n
) b
WHERE a.score > 600
AND a.id = b.id;

**MERGE with only INSERT**

(By default, the config parameter "mergeImplementation" is set to "WITH")

**Input**

MERGE INTO employees01 e
USING (SELECT empid, ename, startdate, address
FROM hr_records
WHERE empid > 100) h
ON (e.id = h.empid)
WHEN NOT MATCHED THEN
INSERT (empid,ename,startdate,address)
VALUES (h.empid,h.ename,h.startdate,h.address);

**Output**

WITH h AS ( SELECT empid, ename, startdate, address
FROM hr_records
WHERE empid > 100
)
INSERT INTO employees01 (empid,ename,startdate,address)
SELECT h.empid, h.ename, h.startdate, h.address
FROM h
WHERE NOT EXISTS ( SELECT 1 FROM UPD_REC e
WHERE e.id = h.empid );

**MERGE with only INSERT**

(If the config parameter "mergeImplementation" is set to "SPLIT")

**Input**

MERGE INTO employees01 e
USING (SELECT empid, ename, startdate, address
FROM hr_records
WHERE empid > 100) h
ON (e.id = h.empid)
WHEN NOT MATCHED THEN
INSERT (empid,ename,startdate,address)
VALUES (h.empid,h.ename,h.startdate,h.address);

**Output**

INSERT INTO employees01 (empid,ename,startdate,address)
SELECT h.empid, h.ename, h.startdate, h.address
FROM ( SELECT empid, ename, startdate, address
FROM hr_records
WHERE empid > 100
) h
LEFT OUTER JOIN employees01 e
ON e.id = h.empid
WHERE e.id IS NULL;

**6.14.4 Pseudo Columns**
6.14.4.1 Overview

This section contains the migration syntax for migrating Oracle (Beta) Pseudo Columns. The migration syntax decides how the supported keywords/features are migrated.

A pseudo column is similar to a table column, but is not actually stored in the table. User can select from pseudo columns, but cannot insert, update, or delete their values.

6.14.4.2 ROWNUM

NOTE

The following limitations are associated with the ROWNUM migration in WHERE clause:

- Only the last ROWNUM will be migrated
- If ROWNUM >= string is present, it will be migrated to limit 0, irrespective of string being a number or function
- If GROUP BY/ORDER BY is present, only the FROM clause containing a single table/subquery will be migrated

Input-ROWNUM in SELECT list

```
SELECT ROWNUM, e.ename, e.empid
FROM employees e
WHERE e.deptno = 10;
```

Output

```
SELECT ROW_NUMBER() OVER() AS ROWNUM, e.ename, e.empid
FROM employees e
WHERE e.deptno = 10;
```

Input-ROWNUM in WHERE clause

```
SELECT e.empid, e.ename FROM employees e WHERE ROWNUM < 6;
```

Output

```
SELECT e.empid, e.ename FROM employees e LIMIT (6 - 1);
```

Input-ROWNUM in WHERE clause with function

```
SELECT X.SOFTASSETS_ID,
X.SOFTASSETS_NAME,
SOFTASSETS_CLASS,
SOFTASSETS_SUBCLASS,
PCKG_SPMS_SYSSOFT_MANAGE.func_dic_value(X.SOFTASSETS_CLASS, '6') SOFTASSETS_CLASS_CHN,
PCKG_SPMS_SYSSOFT_MANAGE.func_dic_value(X.SOFTASSETS_SUBCLASS, '15')
SOFTASSETS_SUBCLASS_CHN,
PCKG_SPMS_SYSSOFT_MANAGE.func_dic_value(X.STATUS, '8')
STATUS,
ROWNUM ROW_ID
FROM (SELECT DISTINCT
SOFTASSETS_ID,
SOFTASSETS_NAME,
SOFTASSETS_CLASS,
SOFTASSETS_SUBCLASS,
STATUS
FROM SPMS_SOFT_ASSETS
WHERE (I_SOFTNAME IS NULL OR
```
(UPPER(SOFTASSETS_NAME) LIKE ('%' || UPPER(I_SOFTNAME) || '%'))
AND SOFTASSETS_CLASS = '002'
AND STATUS = '0'
ORDER BY SOFTASSETS_ID DESC
) X
WHERE ROWNUM < TO_NUMBER
(begNum) + TO_NUMBER (fetchNum);

Output

SELECT X.SOFTASSETS_ID,
    X.SOFTASSETS_NAME,
    SOFTASSETS_CLASS,
    SOFTASSETS_SUBCLASS,
    PCKG_SPMS_SYSSOFT_MANAGE.func_dic_value(X.SOFTASSETS_CLASS, '6')
    SOFTASSETS_CLASS_CHN,
    PCKG_SPMS_SYSSOFT_MANAGE.func_dic_value(X.SOFTASSETS_SUBCLASS, '15')
    SOFTASSETS_SUBCLASS_CHN,
    PCKG_SPMS_SYSSOFT_MANAGE.func_dic_value(X.STATUS, '8') STATUS,
    ROW_NUMBER() OVER(ORDER BY SOFTASSETS_ID) ROW_ID
FROM (SELECT DISTINCT *
    FROM SPMS_SOFT_ASSETS
    WHERE (I_SOFTNAME IS NULL OR (UPPER(SOFTASSETS_NAME) LIKE ('%' ||
    UPPER(I_SOFTNAME) || '%'))
    AND SOFTASSETS_CLASS = '002'
    AND STATUS = '0'
    ORDER BY SOFTASSETS_ID DESC
    ) X
    LIMIT (TO_NUMBER(begNum)+TO_NUMBER(fetchNum) - 1);

Input-ROWNUM with >= Operator and GROUP BY clause

SELECT e.deptno FROM emp e WHERE ROWNUM >= 10 GROUP BY e.deptno;

Output

SELECT e.deptno FROM ( SELECT * FROM emp e LIMIT 0 ) e GROUP BY e.deptno;

Input-ROWNUM in WHERE clause with Additional Condition and ORDER BY Clause

SELECT M.hiredate ,M.ename
FROM ( SELECT ename, hiredate
    FROM emp
    ORDER BY hiredate ) M
WHERE M.hiredate > SYSDATE - 10
    AND ROWNUM < 12
ORDER BY M.hiredate;

Output

SELECT M.hiredate ,M.ename
FROM ( SELECT * FROM ( SELECT ename, hiredate
    FROM emp
    ORDER BY hiredate ) M WHERE M.hiredate > SYSDATE - 10
    LIMIT (12 - 1) ) M
ORDER BY M.hiredate;

Input-ROWNUM in SELECT List and FROM Clause Containing ORDER BY

select a, b, rownum row_id from
( select a,b from t1 order by a
) x where rownum < 20
Y where row_id >=10;

Output
select a, b from (select a, b, ROW_NUMBER() OVER(ORDER BY a) row_id from
select * from t1 order by a
) x LIMIT (20 - 1)
)Y where row_id >=10;

6.14.4.3 ROWID

Input - ROWID without ALIAS

SELECT empid, ename, ROWID FROM employees;

Output

SELECT empid, ename, CAST(( xc_node_id || '#' || tableoid || '#' || ctid ) AS TEXT ) AS ROWID "+
"FROM employees;

Input - ROWID with ALIAS

SELECT empno, ename, e.ROWID AS Row_ID, d.deptno
FROM emp e, dept d
WHERE d.deptno = e.deptno;

Output

SELECT empno, ename, CAST(( e.xc_node_id || '#' || e.tableoid || '#' || e.ctid ) AS TEXT ) AS Row_ID, d.deptno
FROM emp e, dept d
WHERE d.deptno = e.deptno;

Input - Outer Query Having ROWID of Inner Query's Table

SELECT * FROM(
SELECT empno,ename,ROWIDTOCHAR(e.ROWid)
AS groupid,d.deptno
FROM emp e,dept d
WHERE d.deptno = e.deptno);

Output

SELECT * FROM(
SELECT empno,ename,(CAST(( e.xc_node_id || '#' || e.tableoid || '#' || e.ctid ) AS TEXT )) AS groupid,d.deptno
FROM emp e,dept d
WHERE d.deptno = e.deptno);

6.14.5 OUTER JOIN

OUTER QUERY (+)

Join is supported in Gauss 18.2.0. So config parameter is added.

It depends on the following configuration item:

supportJoinOperator=false

Input

SELECT PP.PUBLISH_NO
FROM SPMS_PARAM_PUBLISH PP
WHERE PP.PUBLISH_ID(+) = TB2.PUBLISH_ID;
SELECT  I.APP_CHNAME, I.APP_SHORTNAME
FROM SPMS_APPVERSION SA, SPMS_APP_INFO I
WHERE SA.APP_ID = I.APP_ID(+)
AND SA.DELIVERY_USER = IN_USERID
ORDER BY APPVER_ID DESC ;

Output
SELECT PP.PUBLISH_NO
FROM SPMS_PARAM_PUBLISH PP
WHERE PP.PUBLISH_ID (+) = TB2.PUBLISH_ID;

SELECT I.APP_CHNAME
 , I.APP_SHORTNAME
FROM SPMS_APPVERSION SA
 , SPMS_APP_INFO I
WHERE SA.APP_ID = I.APP_ID (+)
AND SA.DELIVERY_USER = IN_USERID
ORDER BY APPVER_ID DESC;

**OUTER JOIN : 1st_table = 2nd_table (+)**

**Input**

SELECT empno, ename, job, dname, loc
FROM emp, dept
WHERE emp.deptno = dept.deptno (+)
   AND salary > 50000;

**Output**

SELECT empno, ename, job, dname, loc
FROM emp LEFT OUTER JOIN dept
   ON emp.deptno = dept.deptno
WHERE salary > 50000;

**OUTER JOIN : 2nd_table = 1st_table (+)**

**Input**

SELECT empno, ename, job, dname, loc
FROM emp, dept
WHERE dept.deptno = emp.deptno (+)
   AND salary > 50000;

**Output**

SELECT empno, ename, job, dname, loc
FROM dept LEFT OUTER JOIN emp
   ON dept.deptno = emp.deptno
WHERE salary > 50000;

**OUTER JOIN : 1st_table (+) = 2nd_table**

**Input**

SELECT empno, ename, job, dname, loc
FROM emp, dept
WHERE emp.deptno (+) = dept.deptno
   AND salary > 50000;

**Output**

SELECT empno, ename, job, dname, loc
FROM emp RIGHT OUTER JOIN dept
   ON emp.deptno = dept.deptno
WHERE salary > 50000;

**OUTER JOIN : 2nd_table (+) = 1st_table**

**Input**
### SELECT empno, ename, job, dname, loc
FROM emp, dept
WHERE dept.deptno (+) = emp.deptno
AND salary > 50000;

#### Output

SELECT empno, ename, job, dname, loc
FROM dept RIGHT OUTER JOIN emp
ON dept.deptno = emp.deptno
WHERE salary > 50000;

---

### OUTER JOIN using table alias

#### Input

SELECT s.supplier_id, s.supplier_name, o.order_date
FROM suppliers s, orders o
WHERE s.supplier_id = o.supplier_id ( + )
AND s.supplier_id > 1000;

#### Output

SELECT s.supplier_id, s.supplier_name, o.order_date
FROM suppliers s LEFT OUTER JOIN orders o
ON s.supplier_id = o.supplier_id
WHERE s.supplier_id > 1000;

---

### OUTER JOIN with multiple joins and expressions

#### Input

SELECT b.zoneno, b.phybrno, t1.zoneno, t1.brno
FROM schema1.tab1 B, schema2.tab2 t1
WHERE SUBSTR(b.zoneno, 1, 4) = SUBSTR(t1.zoneno, 2, 4) (+)
AND SUBSTR(b.phybrno, 1, 4) = SUBSTR(t1.brno, 2, 4)(+);

#### Output

SELECT b.zoneno, b.phybrno, t1.zoneno, t1.brno
FROM schema1.tab1 B LEFT OUTER JOIN schema2.tab2 t1
ON SUBSTR(b.zoneno, 1, 4) = SUBSTR(t1.zoneno, 2, 4)
AND SUBSTR(b.phybrno, 1, 4) = SUBSTR(t1.brno, 2, 4);

---

### OUTER JOIN with multiple tables and subquery

#### Input

SELECT stru_id, organ1_no, SUBSTR(p_i_date, 1, 8), v_task_date1, TA000
FROM ccm_TAB280362_h d,
( SELECT * FROM tab1 where table_code = 'DA200251007' ORDER BY 1 ) d1,
( SELECT D.dict_cd, D.dict_name
FROM tab2 D WHERE work_dt = SUBSTR(p_i_date, 1, 6) ) d2,
( SELECT * FROM tab3 where table_code = 'DA200251011' ) d3,
( SELECT tab4.dict_cd, tab5.c2
FROM tab4, tab5
WHERE tab5.c1 (+) = tab4.c1
AND table_code = 'DA200251012') d4
WHERE d.TA005 = d1.dict_cd(+)
AND d.TA007 = d2.dict_cd(+)
AND d.TA012 = d3.dict_cd(+)
AND d.TA013 = d4.dict_cd(+)
AND d.TA000 = v_task_date1;
Output

```sql
SELECT stru_id, organ1_no, SUBSTR(p_i_date, 1, 8), v_task_date1, TA000
FROM ccm_TA280362_h d
LEFT OUTER JOIN
( SELECT * FROM tab1 where table_code = 'DA200251007' ORDER BY 1 ) d1
ON d.TA005 = d1.dict_cd
LEFT OUTER JOIN
( SELECT D.dict_cd, D.dict_name FROM tab2 D WHERE work_dt = SUBSTR(p_i_date, 1, 6) ) d2
ON d.TA007 = d2.dict_cd
LEFT OUTER JOIN
( SELECT * FROM tab3 where table_code = 'DA200251011' ) d3
ON d.TA012 = d3.dict_cd
LEFT OUTER JOIN
( SELECT tab4.dict_cd, tab5.c2 FROM tab5 RIGHT OUTER JOIN tab4
WHERE table_code = 'DA200251012' ) d4
ON d.TA013 = d4.dict_cd
WHERE d.TA000 = v_task_date1;
```

OUTER JOIN with static value

Input

```sql
SELECT d.department_name, e.employee_name
FROM departments d, employees e
WHERE d.department_id   = e.department_id (+)
AND e.salary (+)         >= 2000
AND d.department_id >= 30
ORDER BY d.department_name, e.employee_name;
```

Output

```sql
SELECT d.department_name, e.employee_name
FROM departments d LEFT OUTER JOIN employees e
ON d.department_id  = e.department_id
AND e.salary            >= 2000
WHERE d.department_id >= 30
ORDER BY d.department_name, e.employee_name;
```

6.14.6 System Functions

6.14.6.1 Overview

This section contains the migration syntax for migrating Oracle (Beta) system functions. The migration syntax decides how the supported keywords/features are migrated.

6.14.6.2 Date Functions

This section contains information on the Date functions. It is organized as follows:

- **TO_DATE**
- **TIMESTAMP**

**TO_DATE**

In Oracle, TO_DATE converts `char` of CHAR, VARCHAR2, NCHAR, or NVARCHAR2 datatype to a value of DATE datatype.
Input-TO_DATE With Three Parameters

create table AC_T_CLAIMTASKHISTORY
(
    TASKID VARCHAR2(30) not null,
    USERCODE VARCHAR2(10) not null,
    REGISTNO VARCHAR2(22) not null,
    CURRWORKNODE NUMBER(5) default 0,
    POLICYNO VARCHAR2(22) not null,
    CREATETIME DATE,
    TASKENDTIME DATE,
    HASTRANSFER VARCHAR2(1)
)
partition by range (CREATETIME)
(
    partition P0 values less than (TO_DATE(' 1900-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=Gregorian'))
        tablespace TAB_KHYX_ECIF,
    partition PART_2018_01 values less than (TO_DATE(' 2018-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR = Gregorian'))
        tablespace TAB_KHYX_ECIF,
    partition PART_2018_02 values less than (TO_DATE(' 2018-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR = Gregorian'))
        tablespace TAB_KHYX_ECIF,
    partition SYS_P28766 values less than (TO_DATE(' 2018-08-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_calendar=gregorian'))
        tablespace TAB_KHYX_ECIF,
    partition SYS_P29844 values less than (TO_DATE(' 2018-09-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_calendar=Gregorian'))
        tablespace TAB_KHYX_ECIF
);

Output

create table AC_T_CLAIMTASKHISTORY
(
    TASKID VARCHAR2(30) not null,
    USERCODE VARCHAR2(10) not null,
    REGISTNO VARCHAR2(22) not null,
    CURRWORKNODE NUMBER(5) default 0,
    POLICYNO VARCHAR2(22) not null,
    CREATETIME DATE,
    TASKENDTIME DATE,
    HASTRANSFER VARCHAR2(1)
)
partition by range (CREATETIME)
(
    partition P0 values less than (TO_DATE(' 1900-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=Gregorian'))
        tablespace TAB_KHYX_ECIF,
    partition PART_2018_01 values less than (TO_DATE(' 2018-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR = Gregorian'))
        tablespace TAB_KHYX_ECIF,
    partition PART_2018_02 values less than (TO_DATE(' 2018-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR = Gregorian'))
        tablespace TAB_KHYX_ECIF,
    partition SYS_P28766 values less than (TO_DATE(' 2018-08-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_calendar=gregorian'))
        tablespace TAB_KHYX_ECIF,
    partition SYS_P29844 values less than (TO_DATE(' 2018-09-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_calendar=Gregorian'))
        tablespace TAB_KHYX_ECIF
);
For the third parameter in `TO_DATE` function if any value other than `LSCALENDAR=GREGORIAN` appears tool will log an error message.

**NOTE**

**TIMESTAMP**

In Oracle, `TIMESTAMP` stores a combined `DATE` and `TIME` value to be stored. It permits a fractional-seconds value of up to nine digits.

**Input - `TIMESTAMP` with Different Precision**

```sql
CREATE TABLE mc_t_custimportmessage
(
    importid VARCHAR2(30) not null,
    status VARCHAR2(1),
    oldcustid VARCHAR2(30),
    oldcarid VARCHAR2(30),
    carstatus VARCHAR2(2),
    customername VARCHAR2(200),
    identifytype VARCHAR2(30),
    identifynumber VARCHAR2(60),
    phonenumber1 VARCHAR2(50),
    policyno VARCHAR2(66),
    lastciinsuredcompany VARCHAR2(50),
    lastbiinsuredcompany VARCHAR2(50),
    inserttimeforhis TIMESTAMP(9) default sysdate,
    operatetimeforhis TIMESTAMP(9),
    monopolyflag VARCHAR2(3)
);
```

**Output**

```sql
CREATE TABLE mc_t_custimportmessage
(
    importid VARCHAR2(30) not null,
    status VARCHAR2(1),
    oldcustid VARCHAR2(30),
    oldcarid VARCHAR2(30),
    carstatus VARCHAR2(2),
    customername VARCHAR2(200),
    identifytype VARCHAR2(30),
    identifynumber VARCHAR2(60),
    phonenumber1 VARCHAR2(50),
    policyno VARCHAR2(66),
    lastciinsuredcompany VARCHAR2(50),
    lastbiinsuredcompany VARCHAR2(50),
    inserttimeforhis TIMESTAMP(6) default sysdate,
    operatetimeforhis TIMESTAMP(6),
    monopolyflag VARCHAR2(3)
);
```

**6.14.6.3 Other Functions**

**TABLE()**

**Input**

```
SELECT *
FROM
```
### Output

```sql
SELECT *
FROM STAT_DETAIL
WHERE DETAIL_ID IN ( SELECT COLUMN_VALUE
FROM TABLE ( V_DETAIL_ID_ARRAY1 ) A
UNION ALL SELECT COLUMN_VALUE
FROM TABLE ( V_DETAIL_ID_ARRAY2 ) B
UNION ALL SELECT COLUMN_VALUE
FROM TABLE ( V_DETAIL_ID_ARRAY3 ) C )
```

### SQLERRM()

**Input**

```sql
CREATE OR REPLACE EDITIONABLE
PACKAGE BODY PCKG_SQLERRM AS
FUNCTION FUN_SQLERRM (branchId IN VARCHAR2,
languageCode IN VARCHAR2)
RETURN VARCHAR2
IS
roles VARCHAR2 (4000);
BEGIN
SELECT LISTAGG(A_NLS.NAME, ',')
WITHIN GROUP(ORDER BY A_NLS.NAME)
INTO roles
FROM CTP_ROLE A,
CTP_ROLE_NLS A_NLS;
RETURN roles;
EXCEPTION
WHEN OTHERS
THEN
LOG ('FUN_SQLERRM()', SQLERRM (SQLCODE));
RETURN roles;
END FUN_SQLERRM;
```
CREATE OR REPLACE FUNCTION pckg_sqlerrm.FUN_SQLERRM (branchId IN VARCHAR2, languageCode IN VARCHAR2) RETURN VARCHAR2 IS roles VARCHAR2 (4000);
BEGIN
SELECT LISTAGG (A_NLS.NAME, ',') WITHIN GROUP (ORDER BY A_NLS.NAME) INTO roles FROM CTP_ROLE A, CTP_ROLE_NLS A_NLS;
RETURN roles;
EXCEPTION
WHEN OTHERS THEN LOG ('FUN_SQLERRM()', SQLERRM);
RETURN roles;
END;
/

SYSTEM PACKAGE

System package functions like SYS.UTL_FILE() should be migrated as UTIL_FILE().

CREATE OR REPLACE FUNCTION FUNC_ADVANCED_PACKAGE(IN_BINFILE IN SYS.UTL_FILE.FILE_TYPE, IN_CURSOR IN REFCURSOR, IN_PARAM IN NUMBER) RETURN VARCHAR2 IS
V_PARAM1 VARCHAR(32767) := '';
BEGIN
IF IN_CURSOR%ISOPEN THEN
IF IN_PARAM = 1 THEN
V_REC_INDEX := 0;
LOOP
FETCH IN_CURSOR INTO V_PARAM1;
EXIT WHEN IN_CURSOR%NOTFOUND;
SYS.UTL_FILE.PUT_LINE(IN_BINFILE, REPLACE(REPLACE(V_PARAM1,CHR(13),' '),CHR(10),' ') || CHR(13));
END LOOP;
SYS.UTL_FILE.FFLUSH(IN_BINFILE);
CLOSE IN_CURSOR;
ELSE
RETURN '2';
END IF;
END;
CREATE OR REPLACE FUNCTION FUNC_ADVANCED_PACKAGE ( IN_BINFILE IN UTL_FILE.FILE_TYPE, IN_CURSOR IN SYS_REFCURSOR, IN_PARAM IN NUMBER ) RETURN VARCHAR2 IS V_PARAM1 VARCHAR(32767) := '';
BEGIN
  IF IN_CURSOR%ISOPEN THEN
    IF IN_PARAM = 1 THEN
      V_REC_INDEX := 0;
      LOOP
        FETCH IN_CURSOR INTO V_PARAM1;
        EXIT WHEN IN_CURSOR%NOTFOUND;
        UTL_FILE.PUT_LINE ( IN_BINFILE , REPLACE( REPLACE( V_PARAM1 , CHR ( 13 ) , ' ' ) , CHR ( 10 ) , ' ' ) || CHR ( 13 ) ) ;
      END LOOP;
      UTL_FILE.FFLUSH ( IN_BINFILE ) ;
      CLOSE IN_CURSOR;
    ELSE
      RETURN '2' ;
    END IF;
  ELSE
    RETURN '0' ;
  END IF;
EXCEPTION WHEN OTHERS THEN
  RETURN '-1' ;
END ;
/

CREATE OR REPLACE FUNCTION SPLIT ( p_list varchar2, p_sep varchar2 := ';' ) return type_split pipelined is
  l_idx pls_integer;
  v_list varchar2(32767) := p_list;
begin
  loop
    l_idx := instr(v_list, p_sep);
    if l_idx > 0 then
      pipe row(substr(v_list,1,l_idx-1));
      v_list := substr(v_list,l_idx+length(p_sep));
    else
      pipe row(v_list);
      close;
    end if;
  end loop;
  return;
end ;
/

CREATE OR REPLACE FUNCTION "SPLIT" ( p_list varchar2, p_sep varchar2 := ';' ) return type_split pipelined is
  l_idx pls_integer;
  v_list varchar2(32767) := p_list;
begin
  loop
    l_idx := instr(v_list, p_sep);
    if l_idx > 0 then
      pipe row(substr(v_list,1,l_idx-1));
      v_list := substr(v_list,l_idx+length(p_sep));
    else
      pipe row(v_list);
      close;
    end if;
  end loop;
  return;
end ;
/
6.14.7 PL/SQL

6.14.7.1 Overview

This section contains the migration syntax for migrating Oracle PL/SQL. The migration syntax decides how the supported keywords/features are migrated.

PL/SQL is a combination of SQL along with the procedural features of programming languages.

6.14.7.2 END

END with label is not supported in GaussDB 100. So, the label name is removed during migration.

Input - END with procedure name

CREATE PROCEDURE remove_emp (employee_id NUMBER) AS
tot_emps NUMBER;
BEGIN
  DELETE FROM employees
  WHERE employees.employee_id = remove_emp.employee_id;
tot_emps := tot_emps - 1;
END remove_emp;

Output

CREATE PROCEDURE remove_emp (employee_id NUMBER) AS
tot_emps NUMBER;
BEGIN
  DELETE FROM employees
  WHERE employees.employee_id = remove_emp.employee_id;
tot_emps := tot_emps - 1;
END ;

Input - END with function name

CREATE OR REPLACE FUNCTION text_length(a CLOB)
RETURN NUMBER DETERMINISTIC IS
BEGIN
  RETURN getLength(a);
END text_length;

Output

CREATE OR REPLACE FUNCTION text_length(a CLOB)
RETURN NUMBER DETERMINISTIC IS
BEGIN
  RETURN getLength(a);
END ;
CREATE OR REPLACE FUNCTION text_length(a CLOB) RETURN NUMBER DETERMINISTIC IS BEGIN RETURN getLength(a); END;

6.14.7.3 STRING

This is handled by using VARCHAR.

**Input - STRING**

**NOTE**

Indexing a STRING type is not supported.

CREATE OR REPLACE function fn_fnd_var_name ( var_str IN STRING , var_name IN String ) RETURN string IS var_out STRING(10); BEGIN var_out := var_name; DBMS_OUTPUT.PUT_LINE(var_out); RETURN var_out; END fn_fnd_var_name;

**Output**

```
CREATE OR REPLACE function fn_fnd_var_name ( var_str IN VARCHAR , var_name IN VARCHAR ) RETURN VARCHAR IS var_out VARCHAR(10); BEGIN var_out := var_name; DBMS_OUTPUT.PUT_LINE(var_out); RETURN var_out; END fn_fnd_var_name;
```

6.14.7.4 LONG

LONG datatype is migrated as TEXT.

See also: [LONG Data Type in Table Structure](#)

**Input - LONG**

**NOTE**

Conversion inside EXECUTE IMMEDIATE is not supported.

```
cREATE OR REPLACE FUNCTION long_migration_test(samp_text LONG) RETURN LONG AS text_format LONG; BEGIN text_format := 'This is long sample'||samp_text; EXECUTE IMMEDIATE 'CREATE TABLE "LONG" (id long raw, amt LONG)'; text_format := 'UPDATE emp SET ename = "LONG" WHERE empno = :1 RETURNING sal INTO :2'; RETURN text_format; END;
```

**Output**

```
CREATE OR REPLACE FUNCTION long_migration_test(samp_text LONG) RETURN LONG AS text_format LONG; BEGIN text_format := 'This is long sample'||samp_text; EXECUTE IMMEDIATE 'CREATE TABLE "LONG" (id long raw, amt LONG)'; text_format := 'UPDATE emp SET ename = "LONG" WHERE empno = :1 RETURNING sal INTO :2'; RETURN text_format; END;
```
create FUNCTION long_migration_test(samp_text TEXT)
RETURN TEXT
AS
.text_format TEXT;
BEGIN
.text_format := 'This is long sample'||samp_text;
EXECUTE IMMEDIATE 'CREATE TABLE "LONG" (id long raw, amt LONG)';
.text_format := UPDATE emp SET ename = "LONG" WHERE empno = :1 RETURNING sal INTO :2;
RETURN text_format;
END;

6.14.7.5 RESULT_CACHE

When a function with result cache is called, Oracle executes the function, adds the result to the result cache and then returns the function.

When the function call is repeated, Oracle fetches the results from the cache rather than re-execute the function.

Under certain scenarios, this caching behavior can result in significant performance gains.

**Input - RESULT_CACHE**

```sql
CREATE OR REPLACE FUNCTION fn_get_emp_by_eno
(val_in IN NUMBER )
RETURN NUMBER
RESULT_CACHE
IS
l_returnvalue NUMBER;
BEGIN
SELECT deptno
INTO l_returnvalue
FROM emp t
WHERE t.empno = val_in;
RETURN l_returnvalue;
END  fn_get_emp_by_eno;
```

**Output**

```sql
CREATE OR REPLACE FUNCTION fn_get_emp_by_eno
(val_in IN NUMBER )
RETURN NUMBER
IS
l_returnvalue NUMBER;
BEGIN
SELECT deptno
INTO l_returnvalue
FROM emp t
WHERE t.empno = val_in;
RETURN l_returnvalue;
END  fn_get_emp_by_eno;
```

6.14.7.6 Handling NULL constraint

**Input**

```sql
DECLARE
first_name STRING;
last_name STRING NULL;
BEGIN
first_name := 'John '; 
last_name := 'Chen '; 
DBMS_OUTPUT.PUT_LINE('"" || first_name || "");
DBMS_OUTPUT.PUT_LINE('"" || last_name || "");
```

**Output**

```sql
DECLARE
first_name STRING;
last_name STRING NULL;
BEGIN
first_name := 'John '; 
last_name := 'Chen '; 
DBMS_OUTPUT.PUT_LINE('"" || first_name || "");
DBMS_OUTPUT.PUT_LINE('"" || last_name || "");
```
END;
/

DECLARE
first_name VARCHAR;
last_name VARCHAR;
BEGIN
first_name := 'John';
last_name := 'Chen';
DBMS_OUTPUT.PUT_LINE('* ' || first_name || ' *');
DBMS_OUTPUT.PUT_LINE('* ' || last_name || ' *');
END;
/

6.14.8 SQL Plus Commands

In Oracle, the PROMPT command ends the specified message or a blank line to the user’s screen. If you omit text, PROMPT displays a blank line on the user’s screen.

PROMPT

Input
prompt
prompt Creating function ISLEGAL
prompt================================
prompt

Output
\echo prompt
\echo prompt Creating function ISLEGAL
\echo prompt================================
\echo prompt

6.14.9 Chinese Character Support

Input-Chinese ( create table test11(a int,b int)/*create table test11(a int,b int)*/;

Output
create table test11(a int,b int)/*create table test11(a int,b int)*/;

Input-Chinese )
create table test11(a int,b int)/*create table test11(a int,b int)*/;

Output
create table test11(a int,b int)/*create table test11(a int,b int)*/;

Input-Chinese,
create table test11(a int,b int)/*create table test11(a int,b int)*/;

Output
create table test11(a int,b int)/*create table test11(a int,b int)*/;

Input-Support Chinese SPACE
create table test11(a int,b int)/*create table test11(a int,b int)*/;
Output
create table test11(a int,b int); /*create table test11(a int, b int)/;

6.14.10 Nested Comment Support

Input
/*** INSERT INTO tmp_m(ori_carid,carid)
   select /*+ parallel(c,4) *//**
     m.ori_carid, max(c.carid) carid
   from tmp_mcar m, mc_t_basecarinfo c
     where m.frameno = c.frameno
     and m.licenseno = c.licenseno
     and m.comid = c.comid
   AND EXISTS(SELECT 1 FROM TMP_RT_T_FCRA_B d
     WHERE c.carid = d.carid and c.custid = d.custid and c.comid = d.comid and d.enddate >=
     v_startDate and d.enddate < v_endDate )
     and c.validstatus = '1'
     and c.systemflag = '1'
   group by m.ori_carid; ******/

Output
/*** INSERT INTO tmp_m(ori_carid,carid)
   select /*+ parallel(c,4) *//**
     m.ori_carid, max(c.carid) carid
   from tmp_mcar m, mc_t_basecarinfo c
     where m.frameno = c.frameno
     and m.licenseno = c.licenseno
     and m.comid = c.comid
   AND EXISTS(SELECT 1 FROM TMP_RT_T_FCRA_B d
     WHERE c.carid = d.carid and c.custid = d.custid and c.comid = d.comid and d.enddate >=
     v_startDate and d.enddate < v_endDate )
     and c.validstatus = '1'
     and c.systemflag = '1'
   group by m.ori_carid; *******/

6.15 Command Line Reference

6.15.1 Migrate (migrationtool.jar)

Function

migrationtool.jar is used to migrate Teradata and Oracle schemas and queries to DWS.

Format

Linux:

```
sh migrate.sh
   --source-db<source-database>
   --input-folder<input-script-path>
   --output-folder<output-script-path>
   [--application-lang <application-lang>]
   [--migration-type<migration-type>]
   [--log-folder<log-path>]
   [--version-number <Gauss Kernel Version>]
```

Windows:
java -jar migrationtool.jar
--source-db <source-database>
--input-folder <input-script-path>
--output-folder <output-script-path>
[ -application-lang <application-lang> ]
[ --migration-type <migration-type> ]
[ --log-folder <log-path> ]
[ --version-number <Gauss Kernel Version> ]

## Parameter Description

<table>
<thead>
<tr>
<th>Long</th>
<th>Short</th>
<th>Data Type</th>
<th>Description</th>
<th>Value Range</th>
<th>Default Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>--source-db</td>
<td>-S</td>
<td>String</td>
<td>Source database</td>
<td>• Oracle</td>
<td>N/A</td>
<td>--source-db Oracle (or) -S Oracle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Oraclebeta</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Teradata</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--input-folder</td>
<td>-I</td>
<td>String</td>
<td>Input folder containing Teradata or Oracle scripts</td>
<td>N/A</td>
<td>N/A</td>
<td>--input-folder / home/testmigration/Documentation/input (or) -I /home/testmigration/Documentation/input</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--output-folder</td>
<td>-O</td>
<td>String</td>
<td>Output folder where the migrated scripts are saved</td>
<td>N/A</td>
<td>N/A</td>
<td>--output-folder / home/testmigration/Documentation/output (or) -O / home/testmigration/Documentation/output</td>
</tr>
</tbody>
</table>

Table 6-22 Parameters
<table>
<thead>
<tr>
<th>Long</th>
<th>Short</th>
<th>Data Type</th>
<th>Description</th>
<th>Value Range</th>
<th>Default Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>--application-lang</td>
<td>-A</td>
<td>String</td>
<td>Application language parser used for migration</td>
<td>SQL</td>
<td>SQL</td>
<td>--application-lang Perl or -A Perl</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>SQL</strong>: Migrate SQL schemas or scripts in SQL files.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Perl</strong>: Migrate BTEQ or SQL_LANG scripts in Perl files.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--migration-type</td>
<td>-M</td>
<td>String</td>
<td>Migration type Set this parameter based on input scripts.</td>
<td>Bulk</td>
<td>Bulk</td>
<td>--migration-type bulk or -M bulk</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Bulk</strong>: Migrate DML and DDL scripts.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>BLogic</strong>: Migrate service logic, such as procedures and functions.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>BLogic</strong> is used only for Oracle PL/SQL.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--log-folder</td>
<td>-L</td>
<td>String</td>
<td>Log file path</td>
<td>N/A</td>
<td>N/A</td>
<td>--log-folder / home/testmigration/Documentation (or)-L /home/testmigration/Documentation</td>
</tr>
</tbody>
</table>
Usage Guidelines

It is mandatory to specify the source database, input folder path, and output folder path, and optional to specify the migration type and log path.

**NOTE**

If no log path is specified, Migration Tool creates the log folder under TOOL_HOME to store logs.

### Example for Linux

```
sh migrate.sh --source-db Oracle --input-folder
/home/testmigration/Documentation/input--
/output-folder
/home/testmigration/Documentation/output--
/log-folder
/home/testmigration/Documentation--
migration-type bulk
```

### Example in Windows

```
java -jar migrationtool.jar --source-db Oracle --input-folder D:\test\migration\input --output-folder D:\test\migration\output --log-folder D:\test\migration\log --migration-type bulk
```

### System Response

Migration: MigrationTool version initiated by User1
Migration process start time : Tue Aug 08 16:08:06 GMT+08:00 2017
Number of available processors : 4
Configured simultaneous processes in the Tool : 3
33% completed
66% completed
100% completed
**************************************************************************
Total number of SQL files in input folder : 3
Number of queries failed : 0
Number of files failed : 0
Number of files migrated successfully : 3
**************************************************************************
Log file : migrationtool.log is placed in the path : /home/testmigration/Documentation/log
Error Log file : migrationtoolError.log is placed in the path : /home/testmigration/Documentation/log
Migration process end time : Tue Aug 08 16:08:08 GMT+08:00 2017
Total process time : 2680 ms

### 6.15.2 Version

**Function**

The `version` command is used to display the version number of the Migration Tool.

---

<table>
<thead>
<tr>
<th>Long</th>
<th>Short</th>
<th>Data Type</th>
<th>Description</th>
<th>Value Range</th>
<th>Default Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>--version-number</td>
<td>-VN</td>
<td>String</td>
<td>Oraclebeta specified parameter</td>
<td>Oraclebeta</td>
<td>N/A</td>
<td>--version-number or -VN</td>
</tr>
</tbody>
</table>
Format

For Linux:
sh migrate.sh --version | -V

For Windows:
java -jar migrationtool.jar --version

Parameters

None

Usage Guidelines

None

Example for Linux

sh migrate.sh --version

Example for Windows

java -jar migrationtool.jar --version

System Response

Linux
Version: MigrationTool 6.5.1

Windows
Version: Migration (Gauss Tools 6.5.1)

6.15.3 Help

Function

The help command is used to provide the help information for the commands supported by Migration Tool.

Format

Linux:
sh migrate.sh --help | -H

Windows:
java -jar migrationtool.jar --help | -H

Parameter Description

None

Usage Guidelines

None
Example for Linux

sh migrate.sh --help

Example for Windows

java -jar migrationtool.jar --help | -H

6.16 Log Reference

6.16.1 Overview

The log files are the repository for all operations and status of the Migration Tool. The following log files are available:

- **SQL Migration Logs**
  a. migrationtool.log: SQL Migration all activities.
  b. migrationtoolError.log: SQL Migration errors.
  c. successRead.log: SQL Migration successful input file reads.
  d. successWrite.log: SQL Migration successful output file writes.
- **Perl Migration Logs**
  a. perlmigrationtool.log: Perl Migration all activities, warnings and errors.

Apache Log4j is used for the Migration Tool logging framework. The following Log4j configuration files are used and can be customized as required:

- For Teradata/Oracle: config/log4j2.xml
- For Oracle (Beta): config/log4j2_oracle.xml

The topics in this section explain the different logs available for the tool.

6.16.2 SQL Migration Logs

The SQL Migration Tool (migrationtool.jar) supports the following types of logging:

- Activity Logging
- Error Logging
- successRead
- successWrite

**NOTE**

- If a user specifies a log path, then all the logs are saved in the specified log path.
- If a user does not specify the logging path, then the tool creates the log folder in the TOOL_HOME path and saves all the logs in this log folder.
- The maximum size of the log file is 10 MB and users can have maximum of 10 log files, to control disk space.
- The tool does not log sensitive data such as queries.
Activity Logging

Migration Tool saves all log and error information to migrationtool.log. This file is available in the log folder. The migrationtool.log file consists of details such as user who executed the migration and files that have been migrated along with the timestamp. The logging level for activity logging is INFO.

The file structure of the migrationtool.log file is as follows:

2018-09-01 14:54:26,011 INFO CLMigrationUtility:145 MigrationTool is initiated by testmigration
2018-09-01 14:54:26,019 INFO PropertyLoader:98 Successfully loaded Property file : config/application.properties
2018-09-01 14:54:26,019 INFO ApplicationPropertyLoader:36 Application properties have been loaded Successfully
2018-09-01 14:54:26,024 INFO PropertyLoader:98 Successfully loaded Property file : config/features-teradata.properties
2018-09-01 14:54:26,024 INFO FeatureLoader:36 Features have been loaded Successfully
2018-09-01 14:54:26,108 INFO MigrationService:178 Migration process start time : Sat Sep 01 14:54:26 CST 2018
2018-09-01 14:54:26,121 INFO TeradataMigrationService:395 Number of Available Processors: 8
2018-09-01 14:54:26,121 INFO TeradataMigrationService:404 Configured simultaneous processes in the Tool : 3
2018-09-01 14:54:26,122 INFO PropertyLoader:98 Successfully loaded Property file : config/teradata-date-format.properties
2018-09-01 14:54:26,126 INFO MigrationProcessor:95 File name: /home/testmigration/18.1_PI_1_TD/MigrationTool/DTS/DTSffix.sql is started
2018-09-01 14:54:26,129 INFO FileHandler:187 File/home/testmigration/18.1_PI_1_TD/MigrationTool/DTS/DTSffix.sql will be read with charset : us-ascii
2018-09-01 14:54:26,150 INFO FileHandler:704 File/home/testmigration/18.1_PI_1_TD/MigrationTool/DTS/DTSffix.sql will be written with charset : us-ascii
2018-09-01 14:54:26,185 INFO TeradataQueryHandler:71 Position: 1
2018-09-01 14:54:26,218 INFO TeradataQueryHandler:71 Position: 2
2018-09-01 14:54:26,234 INFO FileHandler:704 File/home/testmigration/18.1_PI_1_TD/MigrationTool/DTS/DTSffix.sql will be written with charset : us-ascii
2018-09-01 14:54:26,239 INFO MigrationProcessor:167 File name: /home/testmigration/18.1_PI_1_TD/MigrationTool/DTS/DTSffix.sql is processed successfully
2018-09-01 14:54:26,240 INFO PropertyLoader:323 Successfully updated Property file : config/teradata-date-format.properties
2018-09-01 14:54:26,241 INFO MigrationService:196 Total number of SQL files in Input folder : 1
2018-09-01 14:54:26,241 INFO MigrationService:200 Number of queries failed : 0
2018-09-01 14:54:26,241 INFO MigrationService:203 Number of files failed : 0
2018-09-01 14:54:26,241 INFO MigrationService:208 Number of files migrated successfully : 1
2018-09-01 14:54:26,241 INFO CLMigrationUtility:158 Log file : migrationtool.log and the file is present in the path : /log
2018-09-01 14:54:26,242 INFO CLMigrationUtility:162 Error Log file : migrationtoolError.log and the file is present in the path : /log
2018-09-01 14:54:26,242 INFO CLMigrationUtility:166 Migration process end time : Sat Sep 01 14:54:26 CST 2018
2018-09-01 14:54:26,242 INFO CLMigrationUtility:172 Total process time : 226 ms
2018-09-01 15:34:55,901 INFO CLMigrationUtility:145 MigrationTool is initiated by testmigration
2018-09-01 15:34:55,910 INFO PropertyLoader:98 Successfully loaded Property file : config/application.properties
2018-09-01 15:34:55,911 INFO ApplicationPropertyLoader:36 Application properties have been loaded Successfully
2018-09-01 15:34:56,014 INFO PropertyLoader:98 Successfully loaded Property file: config/teradata-date-format.properties
2018-09-01 15:34:56,019 INFO MigrationProcessor:95 File name: /home/testmigration/18.1_PI_1_TD/MigrationTool/DTS/DTSffix.sql is started
2018-09-01 15:34:56,022 INFO FileHandler:187 File /home/testmigration/18.1_PI_1_TD/MigrationTool/DTS/DTSffix.sql will be read with charset: us-ascii
2018-09-01 15:34:56,043 INFO FileHandler:704 File /home/testmigration/18.1_PI_1_TD/MigrationTool/DTS/DTSffix.sql will be written with charset: us-ascii
2018-09-01 15:34:56,077 INFO TeradataQueryHandler:71 Position: 1
2018-09-01 15:34:56,115 INFO TeradataQueryHandler:71 Position: 2
2018-09-01 15:34:56,133 INFO FileHandler:704 File /home/testmigration/18.1_PI_1_TD/MigrationTool/DTS/DTSffix.sql will be written with charset: us-ascii
2018-09-01 15:34:56,138 INFO MigrationProcessor:167 File name: /home/testmigration/18.1_PI_1_TD/MigrationTool/DTS/DTSffix.sql is processed successfully
2018-09-01 15:34:56,140 INFO PropertyLoader:323 Successfully updated Property file: config/teradata-date-format.properties
2018-09-01 15:34:56,140 INFO MigrationService:196 Total number of SQL files in Input folder: 1
2018-09-01 15:34:56,141 INFO MigrationService:200 Number of queries failed: 0
2018-09-01 15:34:56,141 INFO MigrationService:203 Number of files failed: 0
2018-09-01 15:34:56,141 INFO MigrationService:208 Number of files migrated successfully: 1

Error Logging

Migration Tool logs only the errors that are encountered during the migration process to migrationtoolError.log. This file is available in the log folder. The migrationtoolError.log file consists of details such as date and time of the error and the details of the file (file name) along with the query position. The logging level for error logging is ERROR.

The file structure of the migrationtoolError.log file is as follows:

2017-06-29 14:07:39,585 ERROR TeradataBulkHandler:172 Error occurred during processing of input in Bulk Migration. PreQueryValidation failed in not proper termination or exclude keyword. /home/testmigration/Documentation/Input/c005.sql for Query in position: 4
2017-06-29 14:07:39,962 ERROR TeradataBulkHandler:172 Error occurred during processing of input in Bulk Migration. PreQueryValidation failed in not proper termination or exclude keyword. /home/testmigration/Documentation/Input/c013.sql for Query in position: 11
2017-06-29 14:07:40,136 ERROR QueryConversionUtility:250 Query is not converted as it contains unsupported keyword: join select
2017-06-29 14:07:40,136 ERROR TeradataBulkHandler:172 Error occurred during processing of input in Bulk Migration. PreQueryValidation failed in not proper termination or exclude keyword. /home/testmigration/Documentation/Input/sample.sql for Query in position: 1
2017-06-29 14:07:40,136 ERROR TeradataBulkHandler:172 Error occurred during processing of input in Bulk Migration. PreQueryValidation failed in not proper termination or exclude keyword. /home/testmigration/Documentation/Input/sample.sql for Query in position: 3

Success Read

After a file has been read by the Migration Tool, the file is logged for tracking purposes. In certain scenarios, these logs let the user know the status of the execution of files. This file is available in the log folder. The log file consists of details such as date and time, and the details of the file name. The logging level for this log file is INFO.

The file structure of the successRead.log file is as follows:

2017-07-21 14:13:00,461 INFO readlogger:213 /home/testmigration/Documentation/is not in.sql is read successfully.
2017-07-21 14:13:00,957 INFO readlogger:213 /home/testmigration/Documentation/date quotes.sql is read successfully.
2017-07-21 14:13:01,509 INFO readlogger:213 /home/testmigration/Documentation/column alias replace.sql is read successfully.
Success Write

Migration Tool reads a file, processes it, and writes the output to the disk. This is logged to the success write log file. In some scenarios, this log lets the user know which of the files are successfully processed. In case of a re-run, the user can skip these files and run the remaining files. This file is available in the log folder. The log file consists of details such as date and time, and the details of the file name. The logging level for this log file is INFO.

The file structure of the `successWrite.log` file is as follows:

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-07-21 14:13:00,616</td>
<td>INFO writelogger:595 /home/testmigration/Documentation/is not in.sql has written successfully.</td>
</tr>
<tr>
<td>2017-07-21 14:13:01,055</td>
<td>INFO writelogger:595 /home/testmigration/Documentation/date quotes.sql has written successfully.</td>
</tr>
<tr>
<td>2017-07-21 14:13:01,569</td>
<td>INFO writelogger:595 /home/testmigration/Documentation/column alias replace.sql has written successfully.</td>
</tr>
<tr>
<td>2017-07-21 14:13:02,597</td>
<td>INFO writelogger:213 /home/testmigration/Documentation/2.6BuildInputs/testWithNodataSamples.sql is read successfully.</td>
</tr>
</tbody>
</table>

6.16.3 Perl Migration Logs

The Perl Migration Tool writes all log information to a single file, `perlmigrationtool.log`.

**NOTE**

Since the Perl Migration Tool will execute the SQL Migration Tool to migrate the SQL scripts inside Perl files, the following **SQL migration logs** are also supported:

- Activity Logging
- Error Logging
- successRead
- successWrite

Logging Levels

The logging level for Perl Migration logs is configured using the `logging-level` configuration parameter.

Logging

The Perl Migration Tool saves all log, warnings and error information to `perlmigrationtool.log`. This file is available in the log folder. The log file consists of details such as user who executed the migration and files that have been migrated along with the timestamp.

The structure of the `perlmigrationtool.log` file is as follows:

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-07-08 13:35:10</td>
<td>INFO teradatacore.pm:1316 Extracting SQL contents from perl files started</td>
</tr>
<tr>
<td>2018-07-08 13:35:10</td>
<td>INFO teradatacore.pm:1329 Extracting SQL contents from perl files completed</td>
</tr>
</tbody>
</table>
6.17 Security Management

**NOTICE**

Ensure that the operating system and the required software (refer to [System Requirements](#)) for more details) are updated with the latest patches to prevent vulnerabilities and other security issues.

Security in Migration Tool is managed by access control over the files and folders that are created by the tool. Users must have appropriate permissions to access the files and folders, such as 600/400 for target and log files, 700 for the target and log folders. The tool also ensures data security by not saving sensitive data in the log files.

The file or folder specified in `--input-folder` must not have write privileges to GROUP and/or OTHERS. For security reasons, the tool will not execute if the input files/folders have write privileges.

It is required that the root privileged user must not be used for installation and execution of the Migration Tool for Linux.

The [umask](#) value provided in the [migrate.sh](#) or [migrationtool.jar](#) file is a set value that is related to file permissions. It is recommended that users do not modify this value. Modifying this value will affect file permissions.

**NOTE**

Migration Tool is a standalone application. It does not require any network or database connection to run. It can be run on any machine that is isolated from any network.
6.18 Troubleshooting

This section contains a list of troubleshooting steps and solutions for issues encountered while using Migration Tool.

The table lists the troubleshooting symptoms/issues along with their cause and solution.

Table 6-23 Error Message Reference

<table>
<thead>
<tr>
<th>Symptom/Issue</th>
<th>Cause and Solution</th>
</tr>
</thead>
</table>
| migrate.sh: line 1: export: `file name`: not a valid identifier or The number of input arguments to the application is not expected. Please validate it and try again. | **Cause:** Presence of space in the input folder name.  
**Solution:** Remove space from the input folder name and execute the command again. |
| : command not found migrate.sh: line <line number>: syntax error: unexpected end of file | **Cause:** This could be due to files getting corrupted when copied from Windows to Linux environment. For example, user may be working on systems with dual operating systems. When files are copied from one OS to another, they could get corrupted.  
**Solution:** Run the dos2unix command on the shell script (migrate.sh). |
| Error occurred while formatting! Returning unformatted SQL: select count(*) from table_temp; | **Cause:** This could be due to inconsistency in the number of opening and closing parentheses in the input file.  
**Solution:** Ensure each opening parenthesis has a corresponding closing parenthesis. |
| User1@node79:~/MigrationTool> sh migrate.sh -l SQL_SOURCE -O Target -S Oracle -M Bulk /home/User1/MigrationTool/ Migration: MigrationTool version initiated by User1 Migration Application failed to start: Since the input folder "/home/User1/MigrationTool/SQL_Source" has write privileges to Group and/or Others, process is stopped due to security reason. | **Cause:** Input folder has higher privileges than expected. Hence, process was stopped due to security reasons.  
**Solution:** Execute the following command in the input folder path: chmod 700 <input_folder name> |
<table>
<thead>
<tr>
<th>Symptom/Issue</th>
<th>Cause and Solution</th>
</tr>
</thead>
</table>
| ERROR QueryConversionUtility:249 Query is not converted as it contains unsupported keyword: LAST | **Cause:** Input query file contains an unsupported keyword.  
**Solution:** Ensure no unsupported keywords are present in the scripts to be migrated.  
For details, see **Supported Keywords and Features**. |
| Disk is almost full. Please clear the space and re-run the tool. | **Cause:** Insufficient space on the disk.  
**Solution:** Free space from the disk and retry the operation. |
| Please enter valid input parameters. Kindly refer the user manual to execute. | **Cause:** The possible cause could be:  
1. No valid parameters are specified.  
2. The short keys are in lower case.  
**Solutions:**  
1. Provide all mandatory parameters when performing migration.  
2. Ensure all short keys are in upper case.  
For details, see **Migrate (migrationtool.jar)**. |
| No SQL files found in input folder. Hence stopping migration. | **Cause:** No valid SQL files are present in the input folder during the migration process.  
**Solution:** Ensure SQL files to be migrated are present in the input folder.  
For details, see **Migration Process**. |
| migrate.sh: line 1: syntax error near unexpected token `newline' migrate.sh: line 1: `export TOOL_HOME=<Needs to be updated>' | **Cause:** The TOOL_HOME value is either not set or contains an invalid path.  
**Solution:** Enter the value to be set in TOOL_HOME.  
For more information, refer **4.2 Migration Tool Configuration**. |
| Migration Application failed to start: Currently we are not supporting this Database : <database-name> | **Cause:** Incorrect database name mentioned in the source database parameter.  
**Solution:** Use only Teradata or Oracle as values to the source database parameter.  
For details, see **Migrate (migrationtool.jar)**. |
<table>
<thead>
<tr>
<th>Symptom/Issue</th>
<th>Cause and Solution</th>
</tr>
</thead>
</table>
| Output folder is not set. Please enter an output folder and refer the user manual for syntax. | **Cause:** The output folder path is not specified.  
**Solution:** Specify a valid path for the output folder parameter.  
For details, see Migrate (migrationtool.jar). |
| java.lang.OutOfMemoryError: GC overhead limit exceeded at ... | **Cause:** The system is not able to allocate enough memory to run the Migration Tool.  
**Solution 1:** The preset limit can be modified by changing the Xmx value in migrate.sh. Increase the value for -Xmx in megabytes, for example, -Xmx1024m for 1024 MB.  
For more information, refer to Setting Java Memory All... .  
**Solution 2:** Split the input file to multiple files and run the Migration Tool again. |
### Symptom/Issue

| ascii "****" does not map to charset |

| **Cause:** Migration tool is not able to detect the encoding format of the input file, and the system locale charset does not match the charset of the input file. |

| **Solution:** Set `encodingFormat` to the actual encoding format and try again. |

#### Example:

```bash
testmigration@BLR1000026522:/home/18.1_RETEST/MigrationTool/scripts/teradata> perl sqlTDtoGS.pl -i ../../PERL -o ../../PERL_OUT/ -m /home/testmigration/18.1_FORMAT_RETEST/sep6thpackage/MigrationTool
Extracting SQL contents from perl files started
ascii "\xFF" does not map to Unicode at core/teradatacore.pm line 1270.
ascii "\xFE" does not map to Unicode at core/teradatacore.pm line 1270.
ascii "\xFE" does not map to Unicode at core/teradatacore.pm line 1270.
ascii "\xFF" does not map to Unicode at core/teradatacore.pm line 1270.
Extracting SQL contents from perl files completed
Migrating SQL files
Migrating SQL files completed
Merging migrated SQL contents to perl files started
ascii "\xFF" does not map to Unicode at core/teradatacore.pm line 1723.
ascii "\xFE" does not map to Unicode at core/teradatacore.pm line 1723.
ascii "\xFE" does not map to Unicode at core/teradatacore.pm line 1723.
ascii "\xFF" does not map to Unicode at core/teradatacore.pm line 1723.
Merging migrated SQL contents to perl files completed
Perl file migration completed
*****************************************************************************
********* Effective number of Perl files in input directory : 6
Effective number of Queries failed : 0
Effective number of Perl files failed : 0
Effective number of Perl files migrated successfully : 6
*****************************************************************************
Log files are available at /home/testmigration/18.1_FORMAT_RETEST/sep6thpackage/MigrationTool/log
```

---

### 6.19 FAQs

This section covers the frequently asked questions.

**Q1:** During installation, I get an error "Root privileged users are not allowed to install the migration tool for Linux." What is the solution?
**Answer:** A root privileged user must not be used for installation and execution of the Migration Tool for Linux. It is recommended that a user without root privileges be used to install and operate the migration tool.

**Q2:** How do I configure the Migration Tool to support DWS version V100R002C60 for Teradata?

**Answer:** Perform the following steps to configure the Migration Tool to support DWS version V100R002C60 for Teradata:

1. Open the *features-teradata.properties* file in the *config* subfolder of *TOOL_HOME*.
2. Change the following variable values based on the requirement.
   - **VOLATILE**
   - **PRIMARY INDEX**

   For example,

   ```
   VOLATILE=UNLOGGED / LOCAL TEMPORARY
   PRIMARY INDEX=ONE / MANY
   ```

**NOTE**

The default variable value for **VOLATILE** is *LOCAL TEMPORARY* and for **PRIMARY INDEX** it is *MANY*. 
Chapter 7

Schema Convertor Tool: SQL Syntax Migration Tool

7.1 Introduction to Schema Convertor Tool

7.1.1 Overview

The Schema Convertor Tool is a command line tool that converts MySQL database scripts into DWS database scripts. This topic provides the tool information for all users to learn before using the tool.

Different databases may have different data types, SQL syntax, database objects (such as stored procedures and views), and service logic (such as processes and functions). The process of migrating scripts from one database to another is complex, risky, and time-consuming. However, you can use the Schema Convertor Tool to migrate the scripts from the source database (MySQL) to the DWS data warehouse.

The Schema Convertor Tool does not need to be connected to the database, so you can use it to migrate scripts offline. You only need to prepare the scripts of the source database and place them in the source folder as the input of the Schema Convertor Tool. Then, run the tool to convert the database scripts in the source folder into the DWS database scripts. The tool places the converted scripts in the target folder.

The Schema Convertor Tool migrates scripts to the target database safely and timely, displays the status of the migration process, and records errors that occur during the operation in logs.

It provides a unified interface (migrate.sh/migrate.bat) to migrate the following content from MySQL to the target data warehouse.

- SQL modes
- SQL queries
The preparation of the source database script file must be performed outside the system. You can use some SQL tools to export the source database to an SQL file. The Schema Convertor Tool does not support this function.

7.1.2 System Context

This topic describes the system context of the Schema Convertor Tool. It is a command line tool used by migration engineers to migrate MySQL scripts to the DWS data warehouse. First, the tool reads the SQL script files from the input folder. After the migration is complete, the tool saves the migrated script files to the output folder. Error information generated during tool execution is recorded in the log folder.
7.1.3 System Requirements

This topic describes the software and hardware requirements of the Schema Convertor Tool.

Database Requirements

Source databases supported by Schema Convertor Tool

- MySQL 5.6

Software Requirements

**OS requirements**

The following table lists the OS requirements of the Schema Convertor Tool.

<table>
<thead>
<tr>
<th>OS</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 7</td>
<td>-</td>
</tr>
<tr>
<td>SUSE 11</td>
<td>SP1</td>
</tr>
<tr>
<td></td>
<td>SP2</td>
</tr>
<tr>
<td>OS</td>
<td>Version</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td></td>
<td>SP3</td>
</tr>
<tr>
<td></td>
<td>SP4</td>
</tr>
<tr>
<td>SUSE 12</td>
<td>SP0</td>
</tr>
<tr>
<td></td>
<td>SP1</td>
</tr>
<tr>
<td></td>
<td>SP2</td>
</tr>
<tr>
<td></td>
<td>SP3</td>
</tr>
<tr>
<td>RHEL</td>
<td>6.4</td>
</tr>
<tr>
<td></td>
<td>6.5</td>
</tr>
<tr>
<td></td>
<td>6.6</td>
</tr>
<tr>
<td></td>
<td>6.7</td>
</tr>
<tr>
<td></td>
<td>6.8</td>
</tr>
<tr>
<td></td>
<td>6.9</td>
</tr>
<tr>
<td></td>
<td>7.0</td>
</tr>
<tr>
<td></td>
<td>7.1</td>
</tr>
<tr>
<td></td>
<td>7.2</td>
</tr>
<tr>
<td></td>
<td>7.3</td>
</tr>
<tr>
<td></td>
<td>7.4</td>
</tr>
<tr>
<td>CentOS</td>
<td>6.4</td>
</tr>
<tr>
<td></td>
<td>6.5</td>
</tr>
<tr>
<td></td>
<td>6.6</td>
</tr>
<tr>
<td></td>
<td>6.7</td>
</tr>
<tr>
<td></td>
<td>6.8</td>
</tr>
<tr>
<td></td>
<td>6.9</td>
</tr>
<tr>
<td></td>
<td>7.0</td>
</tr>
<tr>
<td></td>
<td>7.1</td>
</tr>
<tr>
<td></td>
<td>7.2</td>
</tr>
<tr>
<td></td>
<td>7.3</td>
</tr>
</tbody>
</table>

**Other software requirements**

The following table lists the software requirements of the Schema Convertor Tool.
### Table 7-2 Schema Convertor Tool software requirements

<table>
<thead>
<tr>
<th>Software</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>JRE 1.8</td>
<td>Software version used to run the Schema Convertor Tool.</td>
</tr>
</tbody>
</table>

### 7.1.4 Supported MySQL Keywords and Features

This topic describes the keywords and features supported by the MySQL migration tool. The list also provides special implementation methods and configuration parameters of the keywords.

Table 7-3 lists the keywords and features supported by the MySQL migration tool.

### Table 7-3 MySQL keywords and features

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Keyword/Feature</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDL</td>
<td>CREATE DATABASE</td>
<td>V1.0</td>
</tr>
<tr>
<td>DDL</td>
<td>CREATE TABLE</td>
<td>V1.0</td>
</tr>
<tr>
<td>DDL</td>
<td>ALTER TABLE</td>
<td>V1.0</td>
</tr>
<tr>
<td>DDL</td>
<td>CREATE INDEX</td>
<td>V1.0</td>
</tr>
<tr>
<td>DDL</td>
<td>DROP DATABASE</td>
<td>V1.0</td>
</tr>
<tr>
<td>DDL</td>
<td>DROP INDEX</td>
<td>V1.0</td>
</tr>
<tr>
<td>DDL</td>
<td>DROP TABLE</td>
<td>V1.0</td>
</tr>
<tr>
<td>DDL</td>
<td>RENAME TABLE</td>
<td>V1.0</td>
</tr>
<tr>
<td>DDL</td>
<td>TRUNCATE TABLE</td>
<td>V1.0</td>
</tr>
<tr>
<td>DML</td>
<td>DELETE</td>
<td>V1.0</td>
</tr>
<tr>
<td>DML</td>
<td>DO</td>
<td>V1.0</td>
</tr>
<tr>
<td>DML</td>
<td>INSERT</td>
<td>V1.0</td>
</tr>
<tr>
<td>DML</td>
<td>REPLACE</td>
<td>V1.0</td>
</tr>
<tr>
<td>DML</td>
<td>UPDATE</td>
<td>V1.0</td>
</tr>
<tr>
<td>TRANSACTION&amp;LOCK</td>
<td>START TRANSACTION</td>
<td>V1.0</td>
</tr>
<tr>
<td>TRANSACTION&amp;LOCK</td>
<td>COMMIT</td>
<td>V1.0</td>
</tr>
<tr>
<td>TRANSACTION&amp;LOCK</td>
<td>ROLLBACK</td>
<td>V1.0</td>
</tr>
<tr>
<td>TRANSACTION&amp;LOCK</td>
<td>SAVEPOINT</td>
<td>V1.0</td>
</tr>
<tr>
<td>TRANSACTION&amp;LOCK</td>
<td>ROLLBACK TO SAVE SAVEPOINT</td>
<td>V1.0</td>
</tr>
<tr>
<td>Chapter</td>
<td>Keyword/Feature</td>
<td>Version</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>TRANSACTION&amp;LOCK</td>
<td>RELEASE SAVEPOINT</td>
<td>V1.0</td>
</tr>
<tr>
<td>TRANSACTION&amp;LOCK</td>
<td>LOCK TABLES</td>
<td>V1.0</td>
</tr>
<tr>
<td>TRANSACTION&amp;LOCK</td>
<td>UNLOCK TABLES</td>
<td>V1.0</td>
</tr>
<tr>
<td>TRANSACTION&amp;LOCK</td>
<td>SET TRANSACTION</td>
<td>V1.0</td>
</tr>
<tr>
<td>ADMINISTRATION</td>
<td>SET CHARACTER</td>
<td>V1.0</td>
</tr>
<tr>
<td>ADMINISTRATION</td>
<td>CACHE INDEX</td>
<td>V1.0</td>
</tr>
<tr>
<td>ADMINISTRATION</td>
<td>FLUSH</td>
<td>V1.0</td>
</tr>
<tr>
<td>UTILITY</td>
<td>EXPLAIN</td>
<td>V1.0</td>
</tr>
<tr>
<td>UTILITY</td>
<td>USE</td>
<td>V1.0</td>
</tr>
</tbody>
</table>

**NOTE**

- In the output file, an error message is displayed for each unsupported keyword.
- Error messages are recorded in error logs. An error log also contains detailed information about the file, such as the name of the file where the error occurs and the error location. For details, see [Log Reference](#).

### 7.2 Restrictions and Limitations

This topic describes the restrictions and limitations of the Schema Convertor Tool.

**General Restrictions**

- The Schema Convertor Tool is used for syntax migration rather than data migration.

**DWS Database Restrictions and Limitations**

- DWS does not support the 'alias.column name' format.
  ```sql
  UPDATE employees as emp SET emp.name='aaa' WHERE store_id=1;
  ```
- Values of the BLOB type cannot be inserted or updated by running the INSERT or UPDATE statements.

- DWS does not support keyword **REFERENCES**.
  ```sql
  ALTER TABLE EMPLOYEE ADD CONSTRAINT DEPTIDS FOREIGN KEY(DEPT_ID) REFERENCES DEPARTMENT(DEPT_ID);
  ALTER TABLE EMPLOYEE DROP FOREIGN KEY DEPTIDS;
  ALTER TABLE EMPLOYEE ADD CONSTRAINT DEPTIDS FOREIGN KEY(DEPT_ID) REFERENCES DEPARTMENT(DEPT_ID) MATCH FULL;
  ALTER TABLE EMPLOYEE DROP FOREIGN KEY DEPTIDS;
  ALTER TABLE EMPLOYEE ADD CONSTRAINT DEPTIDS FOREIGN KEY(DEPT_ID) REFERENCES DEPARTMENT(DEPT_ID) MATCH PARTIAL ON DELETE CASCADE;
  ```
- DWS does not support the **GeomFromText** and **POINTFromText** functions.
  Examples:
7.3 Schema Convertor Tool Installation

7.3.1 Installing the Schema Convertor Tool

You need to prepare a computer to install the Schema Convertor Tool before using it. Both Windows and Linux OSs are supported.

This topic details the installation process of the Schema Convertor Tool.

Installing the Schema Convertor Tool on a Windows Computer

Step 1 Prepare a computer running the Windows OS.

For details about the software and hardware environment requirements, see System Requirements.

Step 2 Install the JDK 1.8.0 and set the Java path.

Visit the official website of Java to download JDK 1.8.0 or later that matches the OS, and then install the JDK. After the installation succeeds, perform the following steps to check and set the Java path:

1. Choose Start > Control Panel
2. On the Control Panel, click System and Security and System in sequence.
3. In the left pane, click Advanced System Settings. The System Properties dialog box is displayed.
4. Click Environment Variables and check whether JAVA_HOME is correctly set. If JAVA_HOME is not found or is incorrect, click New or Edit to set it.

Step 3 Download the Schema Convertor Tool package. For details, see Downloading Related Tools.

Decompress the downloaded tool package, in which dws-convertor-1.0-SNAPSHOT.zip is applicable to the Windows OS. After decompressing the zip, you can use the Schema Convertor Tool.

----End

Installing the Schema Convertor Tool on a Linux Computer

Step 1 Download the Schema Convertor Tool package. For details, see Downloading Related Tools.

Decompress the downloaded tool package, in which dws-convertor-1.0-SNAPSHOT.tar.gz is applicable to the Linux OS.

Step 2 Prepare a computer running the Linux OS.

For details about the software and hardware environment requirements, see System Requirements.
Step 3 Use WinSCP to upload the dws-convertor-1.0-SNAPSHOT.tar.gz package obtained in Step 1 to the Linux host.

The user who uploads the tool must have the full control permission on the target directory of the Linux host.

Step 4 Use the SSH session tool to remotely log in to the Linux host.

Step 5 Install the JDK 1.8.0 and set the Java path.

Visit the official website of Java to download JDK 1.8.0 or later that matches the OS, and then install the JDK. After the installation succeeds, perform the following steps to check and set the Java path:

1. Run the following command to verify the Java installation:
   ```bash
echo $JAVA_HOME
   ```
2. If no information is returned, run the following command to set the Java path:
   ```bash
   export JAVA_HOME=<Enter Java installation directory>
   
   Example: JAVA_HOME=/home/user/Java/jdk1.8.0_141
   ```
   ```bash
   export PATH=$JAVA_HOME/bin:$PATH
   ```

Step 6 Run the following commands to switch to the path where the Schema Convertor Tool package resides and decompress the package:

```
cd $(Path for storing the Schema Convertor Tool package)
tar -xzf dws-convertor-1.0-SNAPSHOT.tar.gz
```

Step 7 Check the files in the decompressed package.

Run the following command to switch to the Schema Convertor Tool directory:
```
cd dws-convertor-1.0-SNAPSHOT
```

Run the following command to show files in the directory: Ensure that the directory contains the bin, conf, example, and lib folders and the migrate.sh file.
```
ls
```
```
cd bin
ls
```

The dws-convertor-1.0-SNAPSHOT folder contains the files and folders required for executing the Schema Convertor Tool. The following table lists the files and folders:

### Table 7-4 Package content list

<table>
<thead>
<tr>
<th>Folder/File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bin/</td>
<td>-</td>
</tr>
<tr>
<td>bin/migrate.bat</td>
<td>Windows batch processing file, which is used to convert SQL files.</td>
</tr>
<tr>
<td>bin/migrate.sh</td>
<td>Linux Shell script file, which is used to convert SQL files.</td>
</tr>
<tr>
<td>conf/</td>
<td>-</td>
</tr>
<tr>
<td>conf/dws-convertor.properties</td>
<td>Migration configuration file, which is used to configure migration rules for source files.</td>
</tr>
</tbody>
</table>
### 7.3.2 Uninstalling the Schema Convertor Tool

You can uninstall the Schema Convertor Tool by deleting the `dws-convertor-1.0-SNAPSHOT` directory decompressed from the tool package, including all files and folders in it, from the Windows or Linux host on which you installed it.

- `dws-convertor-1.0-SNAPSHOT`
  - `bin/`
  - `conf/`
  - `example/`
  - `lib/`
  - `logs/`

### 7.4 Schema Convertor Tool Configuration

You can define the tool behavior and syntax migration logic during the migration by modifying parameters in the configuration file. This topic describes how to configure the Schema Convertor Tool. You can obtain the expected migration output only after the tool is correctly configured.

Before migrating scripts, you need to configure the tool and migration as required. The following two types of configurations are involved:

1. **Migration configuration**: used to configure the migration rule of the source file. For details about the independent configuration file provided for MySQL, see [Configuring `dws-convertor.properties`].

2. **Log configuration**: used to configure parameters like the log level, log path, and log file size. For details, see [Configuring `log4j.properties`].

**Configuring `dws-convertor.properties`**

By setting application configuration parameters, you can customize the behavior of the migration tool during MySQL database script migration.

Open the `dws-convertor-1.0-SNAPSHOT\conf\dws-convertor.properties` file and configure the parameters in the file as required. **Table 7-5** lists parameters that can be modified.
The parameter values are case-insensitive.
If you configure an incorrect or invalid value for a parameter, the Schema Convertor Tool uses the parameter's default value.

### Table 7-5 Parameters of the `dws-convor.properties` file

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
<th>Default Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>table.database</td>
<td>Default schema</td>
<td>● true</td>
<td>true</td>
<td>table.database AsSchema=true</td>
</tr>
<tr>
<td>table.database AsSchema</td>
<td></td>
<td>● false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>table.defaultSchema</td>
<td>Default schema name</td>
<td>● public</td>
<td>public</td>
<td>table.defaultSchema=public</td>
</tr>
<tr>
<td>table.orientation</td>
<td>Default data storage method. <strong>ROW</strong>: row-based storage; <strong>COLUMN</strong>: column-based storage</td>
<td>● ROW</td>
<td>ROW</td>
<td>table.orientation=ROW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● COLUMN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>table.type</td>
<td>Default table type. <strong>HASH</strong> indicates the partition table, and <strong>REPLICATION</strong> indicates the replication table.</td>
<td>● HASH</td>
<td>HASH</td>
<td>table.type=HASH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● REPLICATION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>table.partition-key.choose.strategy</td>
<td>Policy for selecting the partition key</td>
<td>com.scs.dws.DWSPartitionKeyChooseStrategy</td>
<td>com.scs.dws.DWSPartitionKeyChooserStrategy</td>
<td>table.partition-key.choose.strategy=com.scs.dwspartitionKeyChooserStrategy</td>
</tr>
<tr>
<td>table.compress.mode</td>
<td>Compression algorithm. This parameter is valid only for the row-store tables.</td>
<td>● DELTA</td>
<td>NOCOMPRESS</td>
<td>table.compress.mode=NOCOMPRESS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● PREFIX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>● DICTIONARY</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>● NUMSTR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>● NOCOM PRESS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------</td>
<td>---------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>table.compress.level</td>
<td>Compression level of the table data. It determines the compression ratio and duration. This divides a compression level into sub-levels, providing you with more choices for compression ratio and duration. As the value becomes greater, the compression ratio becomes higher and duration longer at the same compression level.</td>
<td>0-100</td>
<td>0</td>
<td>table.compress.level=0</td>
</tr>
<tr>
<td>table.compress.row</td>
<td>Whether to compress the row-store table. The valid values of the row-store table are YES and NO. The default value is NO.</td>
<td>● YES</td>
<td>NO</td>
<td>table.compress.row=NO</td>
</tr>
<tr>
<td>table.compress.column</td>
<td>Whether to compress the column-store table. The valid values of the column-store table are YES, NO, LOW, MIDDLE, and HIGH. The default value is LOW.</td>
<td>● YES</td>
<td>LOW</td>
<td>table.compress.column=LOW</td>
</tr>
</tbody>
</table>
Configuring log4j.properties

Open the `dws-convertor-1.0-SNAPSHOT\conf\log4j.properties` file. The content is as follows:

```java
log4j.rootLogger=INFO,rca

log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%t] - %m(%F:%L) %n

log4j.appender.rca=org.apache.log4j.RollingFileAppender
log4j.appender.rca.File=../logs/dws-schema-convertor.log
log4j.appender.rca.Encoding=UTF-8
log4j.appender.rca.MaxFileSize=100MB
log4j.appender.rca.MaxBackupIndex=10
log4j.appender.rca.layout=org.apache.log4j.PatternLayout
log4j.appender.rca.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%t] - %m(%F:%L) %n
```

You can configure the following parameters as required. Retain the default settings of other parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value Range</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>log4j.rootLogger</td>
<td>Log Level</td>
<td>OFF, FATAL, ERROR, WARN, INFO, DEBUG, and ALL</td>
<td>INFO</td>
</tr>
<tr>
<td>log4j.appender.rca.Encoding</td>
<td>Sets the method for encoding log files.</td>
<td>UTF-8, GBK, BG2312, UTF-16, and others</td>
<td>UTF-8</td>
</tr>
<tr>
<td>log4j.appender.rca.MaxFileSize</td>
<td>Sets the log file size.</td>
<td>-</td>
<td>100 MB</td>
</tr>
<tr>
<td>log4j.appender.rca.MaxBackupIndex</td>
<td>Sets the total number of log files.</td>
<td>-</td>
<td>10</td>
</tr>
</tbody>
</table>

7.5 Schema Convertor Tool Use
7.5.1 Migration Process Overview

This topic details how to use the Schema Convertor Tool to migrate data.

The Schema Convertor Tool supports the following migration case:

- Migrate MySQL to the DWS data warehouse.

7.5.2 Preparing for Migration

Before migrating scripts, ensure that the following prerequisites are met:

- You have created a data warehouse cluster, which is the target cluster for database script migration.
- You have installed and configured the Schema Convertor Tool.

For detailed operations, see Installing the Schema Convertor Tool and Schema Convertor Tool Configuration.

7.5.3 Obtaining Script Files of the Source Database

Before migration, obtain the script files of the source database (MySQL) first. You can use the SQL client tool to connect to the source database (MySQL) and export the scripts to *.sql files.

On the host where the Schema Convertor Tool is installed, create an input folder and copy all the MySQL database script files to be migrated to this folder. To perform this operation, you must have the write permission on the target folder.

7.5.4 Executing the Schema Convertor Tool

The Schema Convertor Tool (migrate.sh) is a command line tool that can work in offline mode. You do not need to connect the tool to the source or target database. After you run the Schema Convertor Tool, it converts the MySQL database scripts in the input folder into the DWS database scripts and writes the converted scripts to the output folder. The tool supports migration from MySQL to DWS databases, including mode, DDL, DML, Utility, and administration.
Run the following commands to specify the source database, input and output folder paths, log paths, migration language, and migration type:

Command format on the Windows OS:

```
migrate.bat
--source-db <source-database>
--input-folder <input-script-path>
--output-folder <output-script-path>
[--log-folder <log-path>]
[--application-lang <migration-language>]
[--migration-type <migration-type>]
```

Command format on the Linux OS:

```
sh migrate.sh
--source-db <source-database>
--input-folder <input-script-path>
--output-folder <output-script-path>
[--log-folder <log-path>]
[--application-lang <migration-language>]
[--migration-type <migration-type>]
```

For details about the command parameters, see Migrate Command (migrate.sh).

The command parameters are described as follows:

- **source-db**: specifies the source database type. The value is **MySQL**, which is case-insensitive.
- **input-folder**: specifies the path of the input folder for storing the scripts of the source database. The input folder path cannot contain spaces because spaces will cause execution errors for the Schema Convertor Tool.
- **output-folder**: specifies the path of the output folder set before the migration. This parameter is mandatory. The Schema Convertor Tool creates an output folder according to the specified path and stores all migrated files in the folder.
- **log-folder**: specifies the log storage path. This parameter is optional. If no path is not specified, the Schema Convertor Tool automatically creates a log folder in the upper-level directory of **migrate.sh/migrate.bat**. For details about the log, see Log Reference.
- **migration-type**: specifies the migration type. This parameter is optional. The Schema Convertor Tool supports the following migration types. Select one based on the script type.
  - **Bulk**: migrates DML and DDL scripts.
  - **BLogic**: migrates service logic, such as procedures and functions.
Precautions:

- Run the `migrate.sh` command rather than the `.jar` file to execute the Schema Convertor Tool.
- The Schema Convertor Tool migrates only the SQL files in the input folder. Other files are not migrated.
- If the output folder contains sub-folders or files, the Schema Convertor Tool deletes the existing files before the migration. The deleted or overwritten files cannot be restored through the Schema Convertor Tool.
- The Schema Convertor Tool places the migrated files in the output folder. You can compare the files in the input folder with the files after migration.
- If different migration tasks are performed concurrently on the same server (by the same Schema Convertor Tool or different Schema Convertor Tools), different output folder paths and log paths are required.

Example

On the Linux host, run the following command to migrate the MySQL database scripts to the DWS data warehouse: The folder specified in the command is only an example.

```
sh migrate.sh --source-db mysql --input-folder /opt/dws-convertor-1.0-SNAPSHOT/example/input --output-folder /opt/dws-convertor-1.0-SNAPSHOT/example/output --log-folder /opt/dws-convertor-1.0-SNAPSHOT/example/log --application-lang SQL --migration-type bulk
```

The migration details (including the progress and completion status) are displayed on the console.

```
Start to parse insertInto.sql, current index is :1, total size :5
Success to parse insertInto.sql, Cost time :1801ms
Start to parse droptable.sql, current index is :2, total size :5
Success to parse droptable.sql, Cost time :42ms
Start to parse createDatabase.sql, current index is :3, total size :5
Success to parse createDatabase.sql, Cost time :35ms
Start to parse createTable.sql, current index is :4, total size :5
Success to parse createTable.sql, Cost time :26ms
Start to parse deleteTest.sql, current index is :5, total size :5
Success to parse deleteTest.sql, Cost time :5ms
Parse finished, total cost time : 1996ms
Total process time : 2680 ms
```

Total process time indicates the total migration time in ms.

### 7.5.5 Viewing and Verifying the Output Files

**Viewing and Verifying the Output Files**

After the migration process is complete, you can use a comparison tool (for example, BeyondCompare®) to compare the input files with the output files.

Run the following command to view the files in the output folder:

```
    cd OUTPUT
    ls
```

The following information is displayed:

```
total 20
-rw-------. 1 root root 57 Jan 14 07:36 createDatabase.sql
-rw-------. 1 root root 261 Jan 14 07:36 createTable.sql
```

**NOTE**
Viewing Log Files

All operations and error information are written to the corresponding log files. For details, see Log Reference.

7.5.6 Connecting to the DWS Database and Executing the Script Files After Migration

In preceding steps, the source MySQL database scripts have been converted into the DWS database scripts and the converted scripts have been written to the output folder. This topic describes how to connect to the DWS database and execute the script files after migration.

1. Prepare a computer as the DWS client host. Download an SQL client tool and use it to connect to the database in a cluster.

   The Data Studio GUI client is recommended.

   For details about how to install and use Data Studio, see Using the Data Studio GUI Client to Connect to a Cluster.

   To connect to the cluster, obtain the cluster’s connection address, database port, username, password, and database name. You can connect to the default database, for example, postgres.

2. Copy the DWS database script generated by the Schema Convertor Tool to the DWS client host.

3. On the Data Studio interface, choose File > Open from the main menu.

   Alternatively, click Open on the toolbar or right-click SQL Terminal and select Open.

4. In the Open dialog box, select the SQL file to be imported and click Open.

   The Open SQL dialog box is displayed.

5. Click Append to add the SQL script to the SQL Terminal tab page.

   You can also click Overwrite to overwrite the content of the SQL Terminal tab page.

6. Choose Run > Execute Statement to execute the SQL script.

   If the execution fails, locate the fault based on the error information. If you have any technical questions, contact the technical support personnel.

   The Data Studio does not support batch adding and execution of SQL scripts.

7. Repeat steps 3 to 6 to execute all the migrated SQL scripts in sequence.

7.6 Command Line References
7.6.1 Migrate Command (migrate.sh)

Function

The Migrate command is used to migrate the MySQL database modes and queries to the DWS database.

Format

- Command format on the Windows OS:
  ```bash
  migrate.bat
  --source-db <source-database>
  --input-folder <input-script-path>
  --output-folder <output-script-path>
  [--log-folder <log-path>]
  [--application-lang <migration-language>]
  [--migration-type <migration-type>]
  ```

- Command format on the Linux OS:
  ```bash
  sh migrate.sh
  --source-db <source-database>
  --input-folder <input-script-path>
  --output-folder <output-script-path>
  [--log-folder <log-path>]
  [-application-lang <application-lang>]
  [--migration-type <migration-type>]
  ```

Parameter Description

<table>
<thead>
<tr>
<th>Full Name</th>
<th>Abbr</th>
<th>Data Type</th>
<th>Description</th>
<th>Range</th>
<th>Default Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>--source-db</td>
<td>-S</td>
<td>Charater String</td>
<td>Indicates the source database.</td>
<td>MySQL</td>
<td>-</td>
<td>--source-db mysql -S mysql</td>
</tr>
<tr>
<td>--input-folder</td>
<td>-I</td>
<td>Charater String</td>
<td>Indicates the path of the input folder for storing the scripts of the source database (MySQL).</td>
<td>-</td>
<td>-</td>
<td>--input-folder / home/ schemaconverrtortool / example/ input -I /home/ schemaconverrtortool / example/ input</td>
</tr>
<tr>
<td>Full Name</td>
<td>Abbreviation</td>
<td>Data Type</td>
<td>Description</td>
<td>Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------</td>
<td>---------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>--output-folder</td>
<td>-O</td>
<td>Character string</td>
<td>indicates the output folder where migrated scripts are stored.</td>
<td>-</td>
<td>-</td>
<td>• --output-folder / home/schemaconvertortool / example/output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• -O /home/schemaconvertortool / example/output</td>
</tr>
<tr>
<td>--application-lang</td>
<td>-A</td>
<td>Character string</td>
<td>Indicates the migration language parser. This parameter is optional.</td>
<td>SQL</td>
<td>SQL</td>
<td>• --application-lang SQL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SQL: It is used to migrate the SQL modes and scripts in the SQL file.</td>
<td></td>
<td></td>
<td>• -A SQL</td>
</tr>
<tr>
<td>--migration-type</td>
<td>-M</td>
<td>Character string</td>
<td>Indicates the migration type. This parameter is optional.</td>
<td>Bulk</td>
<td>Bulk</td>
<td>• --migration-type bulk</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bulk: It is used to migrate DML and DDL scripts.</td>
<td></td>
<td></td>
<td>• -M bulk</td>
</tr>
<tr>
<td>Full Name</td>
<td>Abbreviation</td>
<td>Data Type</td>
<td>Description</td>
<td>Range</td>
<td>Default Value</td>
<td>Example</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------</td>
<td>---------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>--log-folder</td>
<td>-L</td>
<td>Character string</td>
<td>Indicates the log file path. This parameter is optional.</td>
<td>-</td>
<td>../logs/</td>
<td>• --log-folder /home/schemaconvertortool/example/log</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• -L /home/schemaconvertortool/example/log</td>
</tr>
</tbody>
</table>

**Usage Guideline**

You must specify the source database, and input and output folder paths. The migration type and log path are optional.

**NOTE**

If no log path is specified, the tool creates the log folder in the upper-level directory of the `migrate.sh/migrate.bat` to store all logs.

**Command Example**

```bash
sh migrate.sh --source-db mysql --input-folder /opt/dws-convertor-1.0-SNAPSHOT/example/input --output-folder /opt/dws-convertor-1.0-SNAPSHOT/example/output --log-folder /opt/dws-convertor-1.0-SNAPSHOT/example/log --migration-type bulk
```

**Command Output**

```
Start to parse insertInto.sql, current index is :1, total size :5
Success to parse insertInto.sql, Cost time :1801ms
Start to parse dropTable.sql, current index is :2, total size :5
Success to parse dropTable.sql, Cost time :42ms
Start to parse createDatabase.sql, current index is :3, total size :5
Success to parse createDatabase.sql, Cost time :35ms
Start to parse createTable.sql, current index is :4, total size :5
Success to parse createTable.sql, Cost time :26ms
Start to parse deleteTest.sql, current index is :5, total size :5
Success to parse deleteTest.sql, Cost time :5ms
Parse finished, total cost time : 1996ms
Total process time : 2680 ms
```

**7.6.2 Version Command**

**Function**

The **Version** command is used to display the Schema Convertor Tool version number.

**Format**

```bash
sh migrate.sh --version | -V
```
Parameter Description

None

Usage Guideline

None

Command Example

sh migrate.sh --version

Command Output

Version: DWS Convertor 1.0-SNAPSHOT

7.6.3 Help Command

Function

The `help` command is used to provide the help information for the commands supported by the Schema Convertor Tool.

Format

```
sh migrate.sh --help | -H
```

Parameter Description

None

Usage Guideline

None

Command Example

sh migrate.sh --help

Command Output

```
usage: H
-A,--application-lang
The application language type, which can be either SQL or Perl. Default it SQL. It was a Compatibility parameter in schema convert tool.
-H,--help
Lists for help
-I,--input-folder <arg>
The input/source folder that contains the Teradata/Oracle scripts to be migrated.
-L,--log-folder
The log file path where the log files are created. It was a Compatibility parameter in schema convert tool, the log folder can be changed in log4j.properties.
-M,--migration-type
The migration type, which can be either Bulk or BLogic. It was a Compatibility parameter in schema convert tool.
-O,--output-folder <arg>
The output/target folder where the migrated scripts are placed.
-S,--source-db <arg>
The source database, which can be either MySQL or Oracle.
```
7.7 Log Reference

7.7.1 Overview

Log files are repositories for all operations and statuses of the Schema Convertor Tool. The following log files are supported:

- **SQL migration logs**
  
  *dws-schema-convertor.log*: Records all SQL migration activities.

- **Apache Log4j**
  
  It is used to specify the framework for the Schema Convertor Tool to record logs. You can use the following Log4j configuration file:
  
  - MySQL: *conf/log4j.properties*

This topic describes the logs supported by the tool.

7.7.2 SQL Migration Log

The SQL Schema Convertor Tool (migrate.sh) supports the following types of logs:

- Activity log
- Error log
- Log that records successful read operations
- Log that records successful write operations

**NOTE**

- If you specify a log path, all logs are stored in this path.
- If no log path is specified, the Schema Convertor Tool creates a log folder in the `TOOL_HOME` path to store all logs.
- To control the disk space usage, the maximum size of a log file is 100 MB. You can have a maximum of 10 log files.
- Security-sensitive data shall not be recorded in tool logs.

Activity Log

The Schema Convertor Tool saves all logs and error information to the *dws-schema-convertor.log* file. This file is in the log folder. The *dws-schema-convertor.log* file contains detailed information about users who perform the migration, files to be migrated, and time stamps. The activity log's recording level is INFO.

The structure of the *dws-schema-convertor.log* file is as follows:

```
2018-12-18 03:46:37,269 INFO  [main] - Add class SqlRollbackSavePointUnParser, for dialect DWS, sqlnode SqlRollbackSavePoint.(UnParserFactory.java:49)
2018-12-18 03:46:37,270 INFO  [main] - Add class SqlRootUnParser, for dialect DWS, sqlnode SqlRoot.(UnParserFactory.java:49)
2018-12-18 03:46:37,271 INFO  [main] - Add class SqlSavePointUnParser, for dialect DWS, sqlnode
```
Error Log

The Schema Convertor Tool records only errors that occur during the migration to the `dws-schema-convertor.log` file. This file is in the log folder. The `dws-schema-convertor.log` file contains detailed information about an error, such as the date, time, file details (such as the file name), and query location. The error log's recording level is ERROR.

The structure of the `dws-schema-convertor.log` file is as follows:

2019-01-14 12:28:44.212 ERROR [main] - You have an error in your SQL syntax; check the manual that corresponds to your Database version for the right syntax to use near 'GEOMETRY' at line 198:8
at com.scs.sql.util.ParserException: You have an error in your SQL syntax; check the manual that corresponds to your Database version for the right syntax to use near 'GEOMETRY' at line 198:8
at com.scs.sql.util.ParserException: You have an error in your SQL syntax; check the manual that corresponds to your Database version for the right syntax to use near 'GEOMETRY' at line 198:8
at com.scs.sql.util.ParserException: You have an error in your SQL syntax; check the manual that corresponds to your Database version for the right syntax to use near 'GEOMETRY' at line 198:8
at com.scs.sql.util.ParserException: You have an error in your SQL syntax; check the manual that corresponds to your Database version for the right syntax to use near 'GEOMETRY' at line 198:8
at com.scs.sql.util.ParserException: You have an error in your SQL syntax; check the manual that corresponds to your Database version for the right syntax to use near 'GEOMETRY' at line 198:8
at com.scs.sql.util.ParserException: You have an error in your SQL syntax; check the manual that corresponds to your Database version for the right syntax to use near 'GEOMETRY' at line 198:8
at com.scs.sql.util.ParserException: You have an error in your SQL syntax; check the manual that corresponds to your Database version for the right syntax to use near 'GEOMETRY' at line 198:8

Syntax Tree Parsing

After the Schema Convertor Tool reads the SQL file, the file will be logged for tracking. In some cases, you can obtain the information about the file execution.
status through the log. This file is in the log folder. A log file contains information like the date, time, and file name. The log's recording level is INFO.

The structure of the `dws-schema-convertor.log` file is as follows:

```
2018-12-18 03:46:37,115 INFO  [main] - Parse Completed(MysqlSchemaParser.java:64)
2018-12-18 03:46:37,139 INFO  [main] - visitInsertToTable enter (InsertReplaceStatementVisitor.java:173)
2018-12-18 03:46:37,153 INFO  [main] - visitInsertToTable enter (InsertReplaceStatementVisitor.java:173)
2018-12-18 03:46:37,154 INFO  [main] - visitInsertToTable enter (InsertReplaceStatementVisitor.java:173)
2018-12-18 03:46:37,155 INFO  [main] - visitInsertToTable enter (InsertReplaceStatementVisitor.java:173)
2018-12-18 03:46:37,156 INFO  [main] - visitInsertToTable enter (InsertReplaceStatementVisitor.java:173)
2018-12-18 03:46:37,157 INFO  [main] - visitInsertToTable enter (InsertReplaceStatementVisitor.java:173)
2018-12-18 03:46:37,162 INFO  [main] - visitInsertToTable enter (InsertReplaceStatementVisitor.java:173)
2018-12-18 03:46:37,163 INFO  [main] - visitInsertToTable enter (InsertReplaceStatementVisitor.java:173)
2018-12-18 03:46:37,164 INFO  [main] - visitInsertToTable enter (InsertReplaceStatementVisitor.java:173)
2018-12-18 03:46:37,165 INFO  [main] - visitInsertToTable enter (InsertReplaceStatementVisitor.java:173)
2018-12-18 03:46:37,166 INFO  [main] - visitInsertToTable enter (InsertReplaceStatementVisitor.java:173)
2018-12-18 03:46:37,167 INFO  [main] - visitInsertToTable enter (InsertReplaceStatementVisitor.java:173)
2018-12-18 03:46:37,168 INFO  [main] - visitInsertToTable enter (InsertReplaceStatementVisitor.java:173)
2018-12-18 03:46:37,169 INFO  [main] - visitInsertToTable enter (InsertReplaceStatementVisitor.java:173)
2018-12-18 03:46:37,170 INFO  [main] - visitInsertToTable enter (InsertReplaceStatementVisitor.java:173)
2018-12-18 03:46:37,171 INFO  [main] - visitInsertToTable enter (InsertReplaceStatementVisitor.java:173)
2018-12-18 03:46:37,172 INFO  [main] - visitInsertToTable enter (InsertReplaceStatementVisitor.java:173)
2018-12-18 03:46:37,173 INFO  [main] - visitInsertToTable enter (InsertReplaceStatementVisitor.java:173)
```

SQL File Parsing

The Schema Convertor Tool reads and processes SQL files and writes the output to the disk. This process is recorded in the log file that records successful write operations. In some cases, you can learn files that have been processed successfully through this file. In the case of re-running, you can skip these files and run the remaining files. This file is in the log folder. A log file contains information like the date, time, and file name. The log's recording level is INFO.

The structure of the `dws-schema-convertor.log` file is as follows:

```
2019-01-14 07:36:05,810 INFO  [main] - Parse Completed(MysqlSchemaParser.java:64)
2019-01-14 07:36:05,811 INFO  [main] - visitCreateDatabase enter (DdlStatementVisitor.java:56)
2019-01-14 07:36:05,812 INFO  [main] - SqlCreateDatabase unparse enter(SqlCreateDataBaseUnParser.java:30)
2019-01-14 07:36:05,839 INFO  [main] - SqlCreateDatabase result: CREATE DATABASE "hzq" TEMPLATE=template0 ENCODING='utf8';(SqlCreateDataBaseUnParser.java:58)
2019-01-14 07:36:05,839 INFO  [main] - start to parse sql :
```

7.8 Security Management

---

**NOTICE**

Use the latest patch to update the OS to prevent vulnerabilities and other security issues.

For security purposes, the Schema Convertor Tool controls access to files and folders created by it. If you want to access these files and folders, you must have the required permissions. For example, you need permission 600/400 to access target files and log files, and permission 700 to access the target folder and log folder. In addition, the tool does not store sensitive data in logs to ensure data security.
NOTE

The Schema Convertor Tool is a single-node system application that does not need to connect to any network or database. It can run on any machine that is isolated from any network.
<table>
<thead>
<tr>
<th>Date</th>
<th>What's New</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019-08-08</td>
<td>This is the third official release.</td>
</tr>
<tr>
<td></td>
<td>The following topics are modified:</td>
</tr>
<tr>
<td></td>
<td>● Data Studio - Integrated Database Development Tool</td>
</tr>
<tr>
<td></td>
<td>Upgraded Data Studio to v6.5.1.</td>
</tr>
<tr>
<td></td>
<td>● Migration Tool - SQL Syntax Migration Tool</td>
</tr>
<tr>
<td></td>
<td>Upgraded Migration Tool to v6.5.1.</td>
</tr>
<tr>
<td>2019-06-17</td>
<td>This is the second official release.</td>
</tr>
<tr>
<td></td>
<td>Added:</td>
</tr>
<tr>
<td></td>
<td>● Schema Convertor Tool: SQL Syntax Migration Tool</td>
</tr>
<tr>
<td>2019-05-05</td>
<td>This is the first official release.</td>
</tr>
</tbody>
</table>