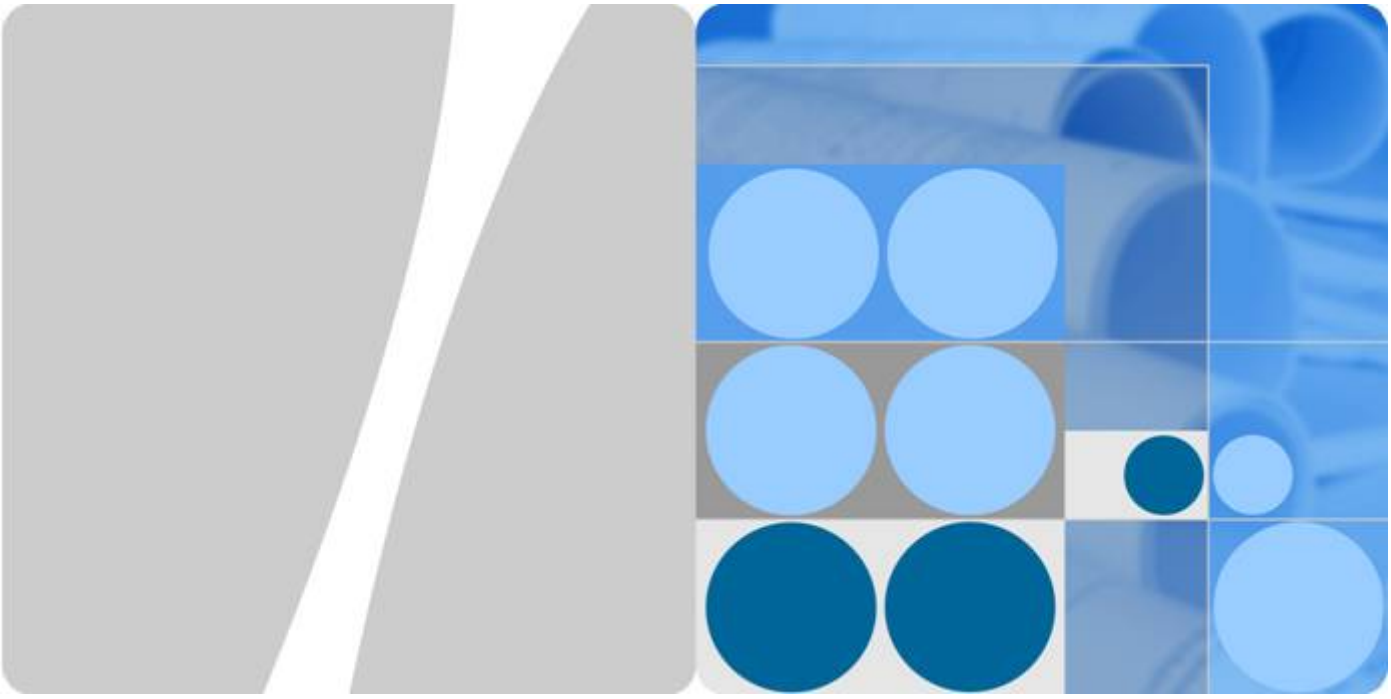


数据仓库服务

工具指南

发布日期　2025-12-11

目 录

1 工具简介

本手册介绍数据仓库服务的工具使用，提供了客户端工具和服务端工具，客户端工具如表1-1所示，服务端工具如表1-2所示。

客户端工具：参见工具下载获取。

服务端工具：位于安装数据库服务器的$GPHOME/script和$GAUSSHOME/bin路径下。

表 1-1 客户端工具

工具名称	工具简介
gsql	一款运行在Linux操作系统的命令行工具，用于连接DWS集群中的数据库，并对数据库进行操作和维护。
Data Studio	用于连接数据库的客户端工具，有着丰富的GUI界面，能够管理数据库和数据库对象，编辑、运行、调试SQL脚本，查看执行计划等。Data Studio工具可运行在32位或64位windows操作系统上，解压软件包后免安装即可使用。
GDS	数据并行加载工具GDS（General Data Service），一款运行在Linux操作系统的命令行工具，通过和外表机制的配合，实现数据的高速导入导出。GDS工具包需要安装在数据源文件所在的服务器上，数据源文件所在的服务器称为数据服务器，也叫GDS服务器。
DSC	DSC（Database Schema Convertor），是一款运行在Linux或Windows操作系统上的命令行工具，用于将Teradata或Oracle数据库中的sql脚本迁移为适用于DWS的sql脚本，便于在DWS中重建数据库。DSC工具是运行在Linux操作系统的命令行工具，解压软件包免安装即可使用。

表 1-2 服务端工具

工具名称	简介
gs_dump	gs_dump是一款用于导出数据库相关信息的工具，支持导出完整一致的数据库对象（数据库、模式、表、视图等）数据，同时不影响用户对数据库的正常访问。
gs_dumpall	gs_dumpall是一款用于导出数据库相关信息的工具，支持导出完整一致的集群数据库所有数据，同时不影响用户对数据库的正常访问。
gs_restore	gs_restore是DWS提供的针对gs_dump导出数据的导入工具。通过此工具可由gs_dump生成的导出文件进行导入。
gds_check	gds_check用于对GDS部署环境进行检查，包括操作系统参数、网络环境、磁盘占用情况等，也支持对可修复系统参数的修复校正，有助于在部署运行GDS时提前发现潜在问题，提高执行成功率。
gds_install	gds_install是用于批量安装gds的脚本工具，可大大提高GDS部署效率。
gds_uninstall	gds_uninstall是用于批量卸载GDS的脚本工具。
gds_ctl	gds_ctl是一个批量控制GDS启停的脚本工具，一次执行可以在多个节点上启动/停止相同端口的GDS服务进程，并在启动时为每一个进程设置看护程序，用于看护GDS进程。

2 工具下载

操作步骤

步骤1 登录DWS控制台。

步骤2 在左侧导航栏中，单击“管理 > 连接客户端”。

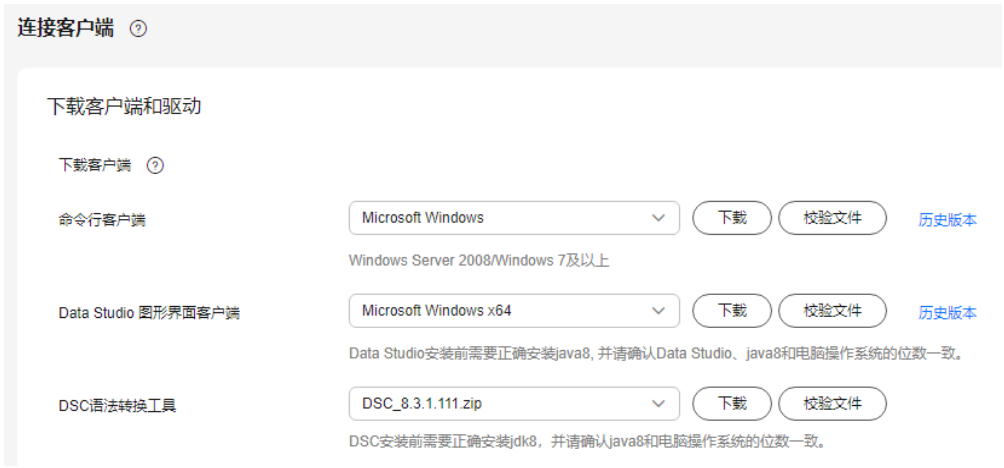
步骤3 在“下载客户端和驱动”区域，请根据计算机的操作系统，选择对应版本的工具进行下载。

此处可以下载以下工具：

- 命令行客户端：gsql工具包中包含了gsql客户端工具、GDS并行数据加载工具以及gs_dump、gs_dumpall和gs_restore工具。
- Data Studio图形界面客户端
- DSC迁移工具

对于gsql客户端工具、Data Studio客户端工具，存在多个历史版本，单击“历史版本”可根据集群版本下载相应版本的工具。DWS 集群可向下兼容gsql、Data Studio工具，建议按集群版本下载配套的工具版本。

图 2-1 下载客户端



----结束

3 gsql

3.1 gsql 概述

基本功能

- **连接数据库：** 通过gsql客户端远程连接DWS数据库。

  说明

  gsql创建连接时，会有5分钟超时时间。如果在这个时间内，数据库未正确地接受连接并对身份进行认证，gsql将超时退出。

  针对此问题，可以参考**常见问题处理**。
- **执行SQL语句：** 支持交互式地键入并执行SQL语句，也可以执行一个文件中指定的SQL语句。
- **执行元命令：** 元命令可以帮助管理员查看数据库对象的信息、查询缓存区信息、格式化SQL输出结果，以及连接到新的数据库等。元命令的详细说明请参见**元命令参考**。

高级特性

gsql的高级特性如**表3-1**所示。

表 3-1 gsql 高级特性

特性名称	描述
变量	gsql提供类似于Linux的shell命令的变量特性，可以使用gsql的元命令\set设置一个变量，格式如下： **\set** *varname value* 要删除一个变量请使用如下方式： **\unset** *varname* **说明** • 变量只是简单的名称/值对，值的长度由特殊变量VAR_MAX_LENGTH决定，详细参见**表3-2**。 • 变量名称必须由字母（包括非拉丁字母）、数字和下划线组成，且对大小写敏感。 • 如果使用\set varname的格式（不带第二个参数），则只是设置这个变量而没有给变量赋值。 • 可以使用不带参数的\set来显示所有变量的值。 变量的示例和详细说明请参见**变量**。
SQL代换	利用gsql的变量特性，可以将常用的SQL语句设置为变量，以简化操作。 SQL代换的示例和详细说明请参见**SQL代换**。
自定义提示符	gsql使用的提示符支持用户自定义。可以通过修改gsql预留的三个变量PROMPT1、PROMPT2、PROMPT3来改变提示符。 这三个变量的值可以用户自定义，也可以使用gsql预定义的值。详细请参见**提示符**。
客户端操作历史记录	gsql支持客户端操作历史记录，当客户端连接时指定“-r”参数，此功能被打开。可以通过\set设置记录历史的条数，例如，\set HISTSIZE 50，将记录历史的条数设置为50，\set HISTSIZE 0，不记录历史。 **说明** • 客户端操作历史记录条数默认设置为32条，最多支持记录500条。当客户端交互式输入包含中文字符时，只支持UTF-8 的编码环境。 • 出于安全考虑，将包含PASSWORD、IDENTIFIED敏感词的记录识别为敏感信息，不会记录到历史信息中，即不能通过上下翻回显。

- 变量

  可以使用gsql元命令\set设置一个变量。例如把变量foo的值设置为bar：

  ```
  \set foo bar
  ```

  要引用变量的值，在变量前面加冒号。例如查看变量的值：

  ```
  \echo :foo
  bar
  ```

  这种变量的引用方法适用于规则的SQL语句和元命令。

  在使用命令行参数--dynamic-param（详见**表3-12**），或设置特殊变量DYNAMIC_PARAM_ENABLE（详见**表3-2**）为true时，可通过执行SQL语句设置变量。变量名为SQL执行结果的列名，也可使用${}方式引用。例如：

  ```
  \set DYNAMIC_PARAM_ENABLE true
  SELECT 'Jack' AS "Name";
   Name
  ------
  ```

```
 Jack
(1 row)

\echo ${Name}
Jack
```

上述示例中，通过SELECT语句执行设置Name变量，并在后面使用${}的引用方式获得变量Name的值。示例中通过特殊变量DYNAMIC_PARAM_ENABLE控制这一功能，也可通过命令行参数--dynamic-param控制，如gsql -d postgres -p 25308 --dynamic-param -r。

说明

- SQL执行失败时，不设置变量。
- SQL执行结果为空，以列名设置变量，赋值空字符串。
- SQL执行结果为一条记录，以列名设置变量，赋值对应字符串。
- SQL执行结果为多条记录，以列名设置变量，使用特定字符串拼接，然后赋值。特定字符串由特殊变量RESULT_DELIMITER（详见**表3-2**）控制，默认为“,”。

执行SQL语句设置变量示例：

```
\set DYNAMIC_PARAM_ENABLE true
CREATE TABLE student (id INT, name VARCHAR(32)) DISTRIBUTE BY HASH(id);
CREATE TABLE
INSERT INTO student VALUES (1, 'Jack'), (2, 'Tom'), (3, 'Jerry');
INSERT 0 3
--  执行失败时，不设置变量
SELECT id, name FROM student ORDER BY idi;
ERROR:  column "idi" does not exist
LINE 1: SELECT id, name FROM student ORDER BY idi;
                                              ^
\echo ${id} ${name}
${id} ${name}

--  执行结果为多条记录时，使用特定字符串拼接
SELECT id, name FROM student ORDER BY id;
 id | name
----+-------
  1 | Jack
  2 | Tom
  3 | Jerry
(3 rows)

\echo ${id} ${name}
1,2,3 Jack,Tom,Jerry

-- 执行结果为一条记录时
SELECT id, name FROM student where id = 1;
 id | name
----+------
  1 | Jack
(1 row)

\echo ${id} ${name}
1 Jack

-- 执行结果为空时，赋值空字符串
SELECT id, name FROM student where id = 4;
 id | name
----+------
(0 rows)

\echo ${id} ${name}
```

gsql预定义了一些特殊变量，同时也规划了变量的取值。为了保证和后续版本最大限度地兼容，请避免以其他目的使用这些变量。所有特殊变量见**表3-2**。

说明

- 所有特殊变量都由大写字母、数字和下划线组成。
- 要查看特殊变量的默认值，请使用元命令**\echo :***varname*（例如\echo :DBNAME）。

表 3-2 特殊变量设置

变量	设置方法	变量说明
DBNAME	\set DBNAME *dbname*	当前连接的数据库的名字。每次连接数据库时都会被重新设置。
ECHO	\set ECHO all | queries	• 如果设置为all，只显示查询信息。设置为all等效于使用gsql连接数据库时指定-a参数。 • 如果设置为queries，显示命令行和查询信息。等效于使用gsql连接数据库时指定-e参数。
ECHO_HIDDEN	\set ECHO_HIDDEN on | off | noexec	当使用元命令查询数据库信息（例如\dg）时，此变量的取值决定了查询的行为： • 设置为on，先显示元命令实际调用的查询语句，然后显示查询结果。等效于使用gsql连接数据库时指定-E参数。 • 设置为off，则只显示查询结果。 • 设置为noexec，则只显示查询信息，不执行查询操作。
ENCODING	\set ENCODING *encoding*	当前客户端的字符集编码。
FETCH_COUNT	\set FETCH_COUNT *variable*	• 如果该变量的值为大于0的整数，假设为n，则执行SELECT语句时每次从结果集中取n行到缓存并显示到屏幕。 • 如果不设置此变量，或设置的值小于等于0，则执行SELECT语句时一次性把结果都取到缓存。 **说明** 设置合理的变量值，将减少内存使用量。一般来说，设为100到1000之间的值比较合理。
HISTCONTROL	\set HISTCONTROL ignorespace | ignoredups | ignoreboth | none	• ignorespace：以空格开始的行将不会写入历史列表。 • ignoredups：与以前历史记录里匹配的行不会写入历史记录。 • ignoreboth、none或者其他值：所有以交互模式读入的行都被保存到历史列表。 **说明** none表示不设置HISTCONTROL。
HISTFILE	\set HISTFILE *filename*	此文件用于存储历史名列表。缺省值是~/.bash_history。

变量	设置方法	变量说明
HISTSIZE	\set HISTSIZE *size*	保存在历史命令里命令的个数。缺省值是500。
HOST	\set HOST *hostname*	已连接的数据库主机名称。
IGNOREEOF	\set IGNOREEOF *variable*	● 若设置此变量为数值，假设为10，则在gsql中输入的前9次EOF字符（通常是Ctrl+C组合键）都会被忽略，在第10次按Ctrl+C才能退出gsql程序。 ● 若设置此变量为非数值，则缺省为10。 ● 若删除此变量，则向交互的gsql会话发送一个EOF终止应用。
LASTOID	\set LASTOID *oid*	最后影响的oid值，即为从一条INSERT或lo_import命令返回的值。此变量只保证在下一条SQL语句的结果显示之前有效。
ON_ERROR_ROLLBACK	\set ON_ERROR_ROLLBACK on | interactive | off	● 如果是on，当一个事务块里的语句产生错误的时候，这个错误将被忽略而事务继续。 ● 如果是interactive，这样的错误只是在交互的会话里忽略。 ● 如果是off（缺省），事务块里一个语句生成的错误将会回滚整个事务。on_error_rollback-on模式是通过在一个事务块的每个命令前隐含地发出一个SAVEPOINT的方式工作的，在发生错误的时候回滚到该事务块。
ON_ERROR_STOP	\set ON_ERROR_STOP on | off	● on：命令执行错误时会立即停止，在交互模式下，gsql会立即返回已执行命令的结果。 ● off（缺省）：命令执行错误时将会跳过错误继续执行。
PORT	\set PORT *port*	正连接数据库的端口号。
USER	\set USER *username*	当前用于连接的数据库用户。
VERBOSITY	\set VERBOSITY terse | default | verbose	这个选项可以设置为值terse、default、verbose之一以控制错误报告的冗余行。 ● terse：仅返回严重且主要的错误文本以及文本位置（一般适合于单行错误信息）。 ● default：返回严重且主要的错误文本及其位置，还包括详细的错误细节、错误提示（可能会跨越多行）。 ● verbose：返回所有的错误信息。

变量	设置方法	变量说明
VAR_NOT_FOUND	\set VAR_NOT_FOUND default | null | error	可以设置为default、null、error之一以控制引用变量不存在时的处理方式。 ● default：不做变量替换，保持原有字符串。 ● null：将原有字符串替换为空字符串。 ● error：输出报错信息，保持原有字符串。
VAR_MAX_LENGTH	\set VAR_MAX_LENGTH *variable*	用于控制变量值的长度，默认为4096。如果变量值的长度超过该值，变量值会被截断，并输出告警信息。
ERROR_LEVEL	\set ERROR_LEVEL transaction | statement	表示ERROR标识成功或者失败类型，取值为transaction或statement，默认为transaction。 ● statement：ERROR记录上一条SQL语句是否执行成功。 ● transaction：ERROR记录上一条SQL语句是否执行成功，或上一个事务内部执行是否出错。
ERROR	\set ERROR true | false	表示上一条SQL语句执行成功或失败，或上一个事务内部执行是否出错，成功取值false，失败取值true，默认为false。执行SQL语句更新，不建议手动设置。
LAST_ERROR_SQLSTATE	\set LAST_ERROR_SQLSTATE state	表示上一条执行失败的SQL语句的错误状态码，默认为‘00000’。执行SQL语句更新，不建议手动设置。
LAST_ERROR_MESSAGE	\set LAST_ERROR_MESSAGE message	表示上一条执行失败的SQL语句的错误信息，默认为空字符串。执行SQL语句更新，不建议手动设置。
ROW_COUNT	\set ROW_COUNT count	● ERROR_LEVEL为statement时，表示上一条SQL语句执行返回的行数或受影响的行数。 ● ERROR_LEVEL为transaction时，如果事务结束时内部有错，表示事务内最后一个SQL语句执行返回的行数或受影响的行数，否则表示上一条SQL语句执行返回的行数或受影响的行数。 如果SQL语句执行失败设置为0，默认为0。执行SQL语句更新，不建议手动设置。

变量	设置方法	变量说明
SQLSTATE	\set SQLSTATE state	• ERROR_LEVEL为statement时，表示上一条SQL语句执行的状态码。 • ERROR_LEVEL为transaction时，如果事务结束时内部有错，表示事务内最后一个SQL语句执行的状态码，否则表示上一条SQL语句执行的状态码。 默认为‘00000’。执行SQL语句更新，不建议手动设置。
LAST_SYS_CODE	\set LAST_SYS_CODE code	表示上一条系统命令执行的返回值，默认为0。使用元命令\!调用系统命令更新，不建议手动设置。
DYNAMIC_PARAM_ENABLE	\set DYNAMIC_PARAM_ENABLE true | false	用于控制执行SQL语句生成变量和${}变量引用方式，默认为false。 • true：执行SQL语句生成变量，支持${}变量引用方式。 • false：执行SQL语句不生成变量，不支持${}变量引用方式。
CONVERT_QUOTE_IN_DYNAMIC_PARAM	\set CONVERT_QUOTE_IN_DYNAMIC_PARAM true | false	用于控制动态变量解析是否需要对单引号、双引号、反斜线进行转义，默认为true。 • true：动态变量解析需要对单引号、双引号、反斜线进行转义，SQL代换时会自动转义变量中的引号和反斜线。 • false：动态变量解析不需要对单引号、双引号、反斜线进行转义，SQL代换时不对变量中的字符串做处理，需要用户根据不同的情况进行手动转义。 使用示例详见**CONVERT_QUOTE_IN_DYNAMIC...**。
RESULT_DELIMITER	\set RESULT_DELIMITER delimiter	执行SQL语句生成变量时，多条记录之间的拼接使用该参数控制，默认为“,”。

变量	设置方法	变量说明
COMPARE_STRATEGY	\set COMPARE_STRATEGY default | natural | equal	用于控制\if表达式中大小比较的策略，默认为default。 • default：默认的比较策略，只支持字符串或数字比较，不支持混合比较。单引号内的按照字符串处理，单引号外的按照数字处理。 • natural：在default的基础上，包含动态变量的按照字符串处理。当比较操作符有一侧是数字，尝试将另一侧转换为数字，然后比较。如果转换失败，报错且比较结果为假。 • equal：只支持等值比较，所有情况按照字符串比较。 详细说明和使用示例见**\if条件块比较规则说明与示例**。
COMMAND_ERROR_STOP	\set COMMAND_ERROR_STOP on | off	用于控制元命令执行错误时是否报错退出，默认不退出。 使用示例详见**COMMAND_ERROR_STOP使用示例**。
INCOMPLETE_QUERY_ERROR	\set INCOMPLETE_QUERY_ERROR true | false	用于控制在\if \goto \for等流程控制元命令执行前是否检测不完整SQL语句，默认为false。 不完整语句包含“(”、“'”、“"”、“$$”不匹配，以及未以分号结尾，检测到后报错退出。使用示例详见**特殊变量INCOMPLETE_QUERY_ERROR使用示例**。
INCOMPLETE_QUERY_ERROR_CODE	\set INCOMPLETE_QUERY_ERROR_CODE 12	在INCOMPLETE_QUERY_ERROR为true时，检测到不完整语句会报错退出，可通过INCOMPLETE_QUERY_ERROR_CODE设置退出码，默认为-1。 使用示例详见**特殊变量INCOMPLETE_QUERY_ERROR_CODE使用示例**。

- 特殊变量ERROR_LEVEL和ERROR使用示例：

当ERROR_LEVEL为statement时，ERROR只记录上一条SQL语句是否执行成功。示例如下，当事务中出现SQL执行报错，事务结束时，ERROR值为false。此时的ERROR只记录上一个SQL语句end是否执行成功。

```
\set ERROR_LEVEL statement
begin;
BEGIN
select 1 as ;
ERROR:  syntax error at or near ";"
LINE 1: select 1 as ;
                    ^
end;
ROLLBACK
```

```
\echo :ERROR
false
```

当ERROR_LEVEL为transaction时，ERROR可以捕获事务内的SQL执行错误。示例如下，事务中出现SQL执行报错，事务结束时，ERROR值为true。

```
\set ERROR_LEVEL transaction
begin;
BEGIN
select 1 as ;
ERROR:  syntax error at or near ";"
LINE 1: select 1 as ;
                    ^
end;
ROLLBACK
\echo :ERROR
true
```

– 特殊变量COMMAND_ERROR_STOP使用示例：

当COMMAND_ERROR_STOP为on时，元命令执行错误时，报错退出。开启时能有效的识别到元命令的执行错误。

当COMMAND_ERROR_STOP为off时，元命令执行错误时，打印相关信息不退出，脚本继续执行。

```
\set COMMAND_ERROR_STOP on
\i /home/omm/copy_data.sql

select id, name from student;
```

如上脚本中COMMAND_ERROR_STOP设置为on，元命令报错之后输出错误信息，脚本不再执行。

```
gsql:test.sql:2: /home/omm/copy_data.sql: Not a directory
```

如果COMMAND_ERROR_STOP设置为off，元命令报错之后输出错误信息，继续执行SELECT语句。

```
gsql:test.sql:2: /home/omm/copy_data.sql: Not a directory
 id | name
----+------
  1 | Jack
(1 row)
```

– 特殊变量INCOMPLETE_QUERY_ERROR使用示例：

当INCOMPLETE_QUERY_ERROR为true时，在\if \goto \for等流程控制元命令前检测不完整SQL语句，并报错退出。

说明

主要识别以下类型不完整语句：

- SQL未以分号结尾。
- SQL中括号不匹配。
- SQL中单引号不匹配。
- SQL中双引号不匹配。
- SQL中$$不匹配。

识别方法：

- 未以分号结尾和括号不匹配：在执行\if \goto \for等流程控制元命令前对字符串分析，剔除C语言风格注释（/**/）和任何空白字符串（包括空格、制表符、换页符等，等价于 [\f\n\r\t\v] ）后还有其他字符，均认为有未完整SQL残留，则报错退出。
- 单引号、双引号和$$不匹配：在未匹配时检测到\if、\elif、\else、\endif、\goto、\label、\for、\loop、\exit-for、\end-for元命令，则报错退出。

SQL未以分号结尾的使用示例：

```
\set INCOMPLETE_QUERY_ERROR true
select 1 as id
```

```
\if ${ERROR}
    \echo 'find error'
    \q 12
\endif
```

如上用例，\if元命令前存在未以分号结尾的SQL语句，在执行\if时会报错退出。

```
$ gsql -X -d postgres -p 13500 --dynamic-param -a -f test.sql
\set INCOMPLETE_QUERY_ERROR true
select 1 as id
\if ${ERROR}
gsql:test.sql:3: ERROR: An incomplete SQL statement exists before the \if command.
gsql:test.sql:3: DETAIL: The SQL statement may not end with a semicolon. Please check.
```

SQL中括号未匹配的使用示例：

```
\set INCOMPLETE_QUERY_ERROR true
insert into student values (1, 'jack';
\if ${ERROR}
    \echo 'find error'
    \q 12
\endif
```

如上用例，\if元命令前存在括号未匹配的SQL语句，在执行\if时会报错退出。

```
$ gsql -X -d postgres -p 13500 --dynamic-param -a -f test.sql
\set INCOMPLETE_QUERY_ERROR true
insert into student values (1, 'jack';
\if ${ERROR}
gsql:test.sql:3: ERROR: An incomplete SQL statement exists before the \if command.
gsql:test.sql:3: DETAIL: There may be an unmatched ( in the SQL statement. Please check.
```

SQL中单引号未匹配的使用示例：

```
\set INCOMPLETE_QUERY_ERROR true
select 'jack as name;
\if ${ERROR}
    \echo 'find error'
    \q 12
\endif
```

如上用例，\if元命令前存在单引号未匹配的SQL语句，在执行\if时会报错退出。

```
$ gsql -X -d postgres -p 13500 --dynamic-param -a -f test.sql
\set INCOMPLETE_QUERY_ERROR true
select 'jack as name;
\if ${ERROR}
gsql:test.sql:3: ERROR: An incomplete SQL statement exists before the \if command.
gsql:test.sql:3: DETAIL: There may be an unmatched ' in the SQL statement. Please check.
```

SQL中双引号未匹配的使用示例：

```
\set INCOMPLETE_QUERY_ERROR true
select 10001 as "ID;
\if ${ERROR}
    \echo 'find error'
    \q 12
\endif
```

如上用例，\if元命令前存在双引号未匹配的SQL语句，在执行\if时会报错退出。

```
$ gsql -X -d postgres -p 13500 --dynamic-param -a -f test.sql
\set INCOMPLETE_QUERY_ERROR true
select 10001 as "ID;
\if ${ERROR}
gsql:test.sql:3: ERROR: An incomplete SQL statement exists before the \if command.
gsql:test.sql:3: DETAIL: There may be an unmatched " in the SQL statement. Please check.
```

SQL中$$未匹配的使用示例：

```
\set INCOMPLETE_QUERY_ERROR true
create or replace function gsql_dollar_quote_test()
returns integer
as
$BODY$
declare
    query text;
```

```
    dest text;
begin
    query := 'select count(*) from pg_class';
    execute immediate query into dest;
end;
$BODY
language 'plpgsql' not fenced;
call gsql_dollar_quote_test();
\if ${ERROR}
    \echo 'find error'
    \q 12
\endif
```

如上用例，\if元命令前存在$$未匹配的SQL语句，在执行\if时会报错退出。

```
$ gsql -X -d postgres -p 13500  --dynamic-param -a -f test.sql
\set INCOMPLETE_QUERY_ERROR true
create or replace function gsql_dollar_quote_test()
returns integer
as
$BODY$
declare
    query text;
    dest text;
begin
    query := 'select count(*) from pg_class';
    execute immediate query into dest;
end;
$BODY
language 'plpgsql' not fenced;
call gsql_dollar_quote_test();
\if ${ERROR}
gsql:test.sql:16: ERROR: An incomplete SQL statement exists before the \if command.
gsql:test.sql:16: DETAIL: There may be an unmatched $$ in the SQL statement. Please check.
```

– 特殊变量INCOMPLETE_QUERY_ERROR_CODE使用示例：

当INCOMPLETE_QUERY_ERROR为true，可通过INCOMPLETE_QUERY_ERROR_CODE设置检测到不完整语句时的退出码。使用示例如下：

```
\set INCOMPLETE_QUERY_ERROR true
\set INCOMPLETE_QUERY_ERROR_CODE 20
insert into student values (1, 'jack';
\if ${ERROR}
    \echo 'find error'
    \q 12
\endif
```

如上用例，设置INCOMPLETE_QUERY_ERROR_CODE为20，则\if检测到不完整语句后退出，且退出码为INCOMPLETE_QUERY_ERROR_CODE的值。

```
$ gsql -X -d postgres -p 13500  --dynamic-param -a -f test.sql
\set INCOMPLETE_QUERY_ERROR true
\set INCOMPLETE_QUERY_ERROR_CODE 20
insert into student values (1, 'jack';
\if ${ERROR}
gsql:test.sql:4: ERROR: An incomplete SQL statement exists before the \if command.
gsql:test.sql:4: DETAIL: There may be an unmatched ( in the SQL statement. Please check.
$ echo $?
20
```

- SQL代换

  像元命令的参数一样，gsql变量的一个关键特性是可以把gsql变量替换成正规的SQL语句。此外，gsql还提供为变量更换新的别名或其他标识符等功能。使用SQL代换方式替换一个变量的值可在变量前加冒号。例如：

```
\set foo 'HR.areaS'
select * from :foo;
 area_id |       area_name
---------+------------------------
       4 | Iron
```

```
      3 | Desert
      1 | Wood
      2 | Lake
(4 rows)
```

执行以上命令，将会查询HR.areaS表。

须知

变量的值是逐字复制的，甚至可以包含不对称的引号或反斜杠命令。所以必须保证输入的内容有意义。

- 特殊变量CONVERT_QUOTE_IN_DYNAMIC_PARAM使用示例：

当CONVERT_QUOTE_IN_DYNAMIC_PARAM为true时，SQL代换时会自动转义变量中的引号和反斜线。

```
 \set DYNAMIC_PARAM_ENABLE true
 \set CONVERT_QUOTE_IN_DYNAMIC_PARAM true
 select '""abc""\\' as "SpecialCharacters";
   test
------------
 ""abc""\\
(1 row)

-- 单引号转义，结果中还是两个单引号
 select '${SpecialCharacters}' as "test";
   test
------------
 ""abc""\\
(1 row)

-- 单引号、反斜线转义，结果中还是两个单引号、两个反斜线
 select E'${SpecialCharacters}' as "test";
   test
------------
 ""abc""\\
(1 row)

-- 双引号转义，结果中还是两个单引号
-- 因为列名中有字母、数字、下划线之外的其他字符，所以有错误信息
 select 'test' as "${SpecialCharacters}";
error while saving the value of ""abc""\\, please check the column name which can only contain upper and lower case letters, numbers and '_'.
 ""abc""\\
------------
 test
(1 row)
```

当CONVERT_QUOTE_IN_DYNAMIC_PARAM为false时，SQL代换时不对变量中的字符串做处理，需要用户根据不同的情况进行手动转义。

说明

不建议用户设置CONVERT_QUOTE_IN_DYNAMIC_PARAM为false，建议使用默认的true。

因为SQL代换时，''内需要对单引号转义，E''内需要对单引号、反斜线转义，""内需要对双引号转义。用户需要根据变量所在的位置不同，对引号和反斜线进行不同的处理。这使得SQL代换中变量使用逻辑复杂且易出错。

```
 \set DYNAMIC_PARAM_ENABLE true
 \set CONVERT_QUOTE_IN_DYNAMIC_PARAM false
 select '""abc""\\' as "SpecialCharacters";
   test
------------
 ""abc""\\
(1 row)
```

```
-- 单引号未转义，结果中只有一个单引号
 select '${SpecialCharacters}' as "test";
   test
----------
 ""abc'\\
(1 row)

-- 单引号、反斜线未转义，结果中只有一个单引号、一个反斜线
 select E'${SpecialCharacters}' as "test";
  test
---------
 ""abc'\
(1 row)

-- 双引号未转义，结果中只有一个双引号
-- 因为列名中有字母、数字、下划线之外的其他字符，所以有错误信息
 select 'test' as "${SpecialCharacters}";
error while saving the value of "abc"\\, please check the column name which can only contain upper and lower case letters, numbers and '_'.
 "abc"\\
----------
 test
(1 row)
```

- 提示符

  通过表3-3的三个变量可以设置gsql的提示符，这些变量是由字符和特殊的转义字符所组成。

表 3-3 提示符变量

变量	描述	示例

PROMPT1	gsql请求一个新命令时使用的正常提示符。
	PROMPT1的默认值为：
	`%/%R%#`
		使用变量PROMPT1切换提示符：
		• 提示符变为[local]：
		`\set PROMPT1 %M`
		`[local:/tmp/gaussdba_mppdb]`
		• 提示符变为name：
		`\set PROMPT1 name`
		`name`
		• 提示符变为=：
		`\set PROMPT1 %R`
		`=`

PROMPT2	在一个命令输入期待更多输入时（例如，查询没有用一个分号结束或者引号不完整）显示的提示符。	使用变量PROMPT2显示提示符：
		`\set PROMPT2 TEST`
		`select * from HR.areaS TEST;`
		` area_id |  area_name`
		`---------+--------------------`
		`  1 | Wood`
		`  2 | Lake`
		`  4 | Iron`
		`  3 | Desert`
		`(4 rows))`

PROMPT3	当执行COPY命令，并期望在终端输入数据时（例如，COPY FROM STDIN），显示提示符。	使用变量PROMPT3显示COPY提示符：
		`\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`
		`>>>>\.`

提示符变量的值是按实际字符显示的，但是，当设置提示符的命令中出现“%”时，变量的值根据“%”后的字符，替换为已定义的内容，已定义的提示符请参见表3-4。

表 3-4 已定义的替换

符号	符号说明
%M	主机的全名（包含域名），若连接是通过Unix域套接字进行的，则全名为[local]，若Unix域套接字不是编译的缺省位置，就是[local:/dir/name]。
%m	主机名删去第一个点后面的部分。若通过Unix域套接字连接，则为[local]。
%>	主机正在侦听的端口号。
%n	数据库会话的用户名。
%/	当前数据库名称。
%~	类似 %/，如果数据库是缺省数据库时输出的是波浪线~。
%#	如果会话用户是数据库系统管理员，使用#，否则用>。
%R	• 对于PROMPT1通常是“=”，如果是单行模式则是“^”，如果会话与数据库断开（如果\connect失败可能发生）则是“!”。 • 对于PROMPT2该序列被“ -”、“ *”、单引号、双引号或“$”(取决于gsql是否等待更多的输入：查询没有终止、正在一个 /* ... */ 注释里、正在引号或者美元符扩展里)代替。
%x	事务状态： • 如果不在事务块里，则是一个空字符串。 • 如果在事务块里，则是“*”。 • 如果在一个失败的事务块里则是“!”。 • 如果无法判断事务状态时为“?”（比如没有连接）。
%digits	指定字节值的字符将被替换到该位置。
%:name	gsql变量“name”的值。
%command	command的输出，类似于使用“^”替换。
%[. . . %]	提示可以包含终端控制字符，这些字符可以改变颜色、背景、提示文本的风格、终端窗口的标题。例如， postgres=> \set PROMPT1 '%[%033[1;33;40m%]%n@%/%R%[%033[0m%]%#' 这个句式的结果是在VT100兼容的可显示彩色的终端上的一个宽体（1;）黑底黄字（33;40）。

环境变量

表 3-5 与 gsql 相关的环境变量

名称	描述
COLUMNS	如果\set columns为0，则由此参数控制wrapped格式的宽度。这个宽度用于决定在自动扩展的模式下，是否要把宽输出模式变成竖线的格式。
PAGER	如果查询结果无法在一页显示，它们就会被重定向到这个命令。可以用\pset命令关闭分页器。典型的是用命令more或less来实现逐页查看。缺省值是平台相关的。 **说明** less的文本显示，受系统环境变量LC_CTYPE影响。
PSQL_EDITOR	\e和\ef命令使用环境变量指定的编辑器。变量是按照列出的先后顺序检查的。在Unix系统上默认的编辑工具是vi。
EDITOR	
VISUAL	

PSQL_EDITOR_LINENUMBER_ARG	当\e和\ef带上一行数字参数使用时，这个变量指定的命令行参数用于向编辑器传递起始行数。像Emacs或vi这样的编辑器，这只是个加号。如果选项和行号之间需要空白，在变量的值后加一个空格。例如：
	`PSQL_EDITOR_LINENUMBER_ARG = '+'`
	`PSQL_EDITOR_LINENUMBER_ARG='--line '`
	Unix系统默认的是+。

PSQLRC	用户的.gsqlrc文件的交互位置。
SHELL	使用\!命令跟shell执行的命令是一样的效果。
TMPDIR	存储临时文件的目录。缺省是/tmp。

3.2 下载客户端

DWS提供了与集群版本配套的客户端工具包，用户可以在DWS管理控制台下载客户端工具包。

客户端工具包包含以下内容：

- **数据库连接工具Linux gsql和测试样例数据的脚本**

  Linux gsql是一款运行在Linux环境上的命令行客户端，用于连接DWS集群中的数据库。

  测试样例数据的脚本是执行入门示例时用的。

- **Windows版本gsql**

  Windows gsql是一款运行在Windows环境上的命令行客户端，用于连接DWS集群中的数据库。

  说明

  仅8.1.3.101及以上集群版本支持在console控制台下载。

- **GDS工具包**

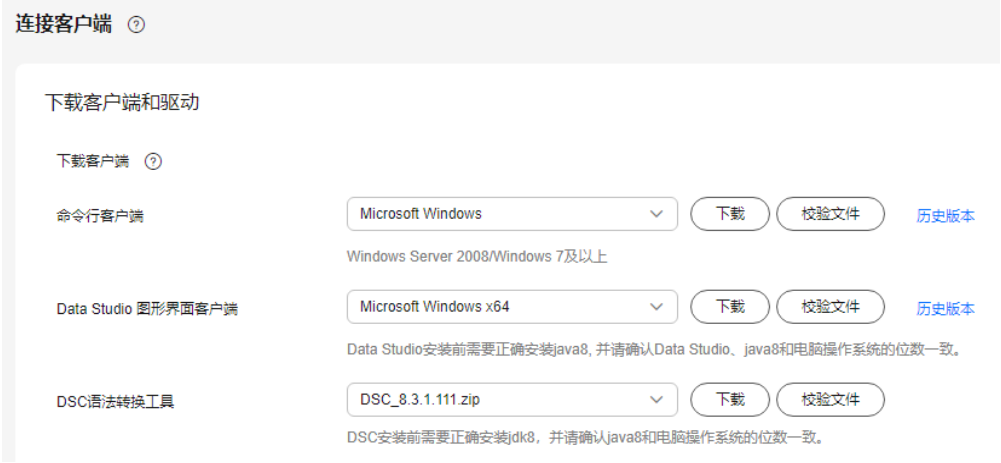
  GDS工具包是数据服务工具。用户可以使用GDS工具将普通文件系统中的数据文件导入到DWS数据库中，GDS工具包需要安装在数据源文件所在的服务器上。数据源文件所在的服务器称为数据服务器，也称为GDS服务器。

下载客户端

步骤1 登录DWS控制台，在左侧导航栏中，单击“连接管理”。

步骤2 在“gsql命令行客户端”的下拉列表中，选择对应版本的DWS客户端。

图 3-1 下载客户端



步骤3 单击“下载”可以下载与8.1.x集群版本匹配的gsql。单击“历史版本”可根据集群版本下载相应版本的gsql。

- 推荐下载使用与集群版本匹配的gsql工具，即8.1.0及以上版本集群使用8.1.x版本gsql、8.2.0及以上版本集群使用8.2.x版。
- 表3-6列出了下载的Linux gsql工具包中的文件和文件夹。

表 3-6 Linux gsql 工具包目录及文件说明

文件或文件夹	说明
bin	该文件夹中包含了gsql在Linux中的可执行文件。其中包含了gsql客户端工具、GDS并行数据加载工具以及gs_dump、gs_dumpall和gs_restore工具。
gds	该文件夹中包括了GDS数据服务工具的相关文件，GDS工具用于并行数据加载，可将存储在普通文件系统中的数据文件导入到DWS数据库中。
lib	该文件夹中包括执行gsql所需依赖的lib库。

文件或文件夹	说明
sample	该文件夹中包含了以下目录或文件： – setup.sh：在使用gsql导入样例数据前所需执行的配置AK/SK访问密钥的脚本文件。 – tpcds_load_data_from_obs.sql：使用gsql客户端导入TPC-DS样例数据的脚本文件。 – query_sql目录：查询TPC-DS样例数据的脚本文件。
gsql_env.sh	在运行gsql前，配置环境变量的脚本文件。

- 表3-7列出了下载的Windows gsql工具包中的文件和文件夹。

  表 3-7 Windows gsql 工具包目录及文件说明

文件或文件夹	说明
x64	该文件夹中包含了64位Windows gsql执行二进制和动态库。
x86	该文件夹中包含了32位Windows gsql执行二进制和动态库。

  说明

  – 在“集群管理”页面的集群列表中，单击指定集群的名称，再选择“集群详情”页签，可查看集群版本。

----结束

3.3 使用指导

3.3.1 使用 Linux gsql 客户端连接集群

用户在创建好数据仓库集群，开始使用集群数据库之前，需要使用数据库SQL客户端连接到数据库。DWS提供了与集群版本配套的Linux gsql命令行客户端工具，您可以使用Linux gsql客户端通过集群的公网地址或者内网地址访问集群。

它的运行环境是Linux操作系统，在使用Linux gsql客户端远程连接DWS集群之前，需要准备一个Linux主机用于安装和运行Linux gsql客户端。如果通过公网地址访问集群，也可以将Linux gsql客户端安装在用户自己的Linux主机上，但是该Linux主机必须具有公网地址。若DWS集群没有配置公网IP，为方便起见，推荐您创建一台Linux弹性云服务器（简称ECS），详情可参见**（可选）准备ECS作为gsql客户端主机**。

（可选）准备 ECS 作为 gsql 客户端主机

创建弹性云服务器的操作步骤，请参见《弹性云服务器用户指南》中的“快速入门 > 创建弹性云服务器”章节。

创建的弹性云服务器需要满足如下要求：

- 弹性云服务器需要与DWS集群具有相同的区域、可用区。
- 如果使用DWS提供的gsql命令行客户端连接DWS集群，弹性云服务器的镜像必须满足如下要求：

  镜像的操作系统必须是gsql客户端所支持的下列Linux操作系统：
  - “Redhat x86_64”客户端工具支持在以下系统中使用：
    - RHEL 6.4~7.6。
    - CentOS 6.4~7.4。
    - EulerOS 2.3。
  - “SUSE x86_64”客户端工具支持在以下系统中使用：
    - SLES 11.1~11.4。
    - SLES 12.0~12.3。
  - “Euler Kunpeng_64”客户端工具支持在以下系统中使用：
    - EulerOS 2.8。
- 如果客户端通过内网地址访问集群，请确保创建的弹性云服务器与DWS集群在同一虚拟私有云里。

  虚拟私有云相关操作请参见《虚拟私有云用户指南》中“虚拟私有云和子网”。
- 如果客户端通过公网地址访问集群，请确保创建的弹性云服务器和DWS集群都要有弹性IP。

  创建弹性云服务器时，参数“弹性IP”需设置为“自动分配”或“使用已有”。
- 弹性云服务器对应的安全组规则需要确保能与DWS集群提供服务的端口网络互通。

  安全组相关操作请参见《虚拟私有云用户指南》中“安全组”章节。

  请确认弹性云服务器的安全组中存在符合如下要求的规则，如果不存在，请在弹性云服务器的安全组中添加相应的规则：
  - 方向：出方向
  - 协议：必须包含TCP，例如TCP、全部。
  - 端口：需要包含DWS集群提供服务的数据库端口，例如，设置为“1-65535”或者具体的DWS数据库端口。
  - 目的地址：设置的IP地址需要包含所要连接的DWS集群的连接地址。其中0.0.0.0/0表示任意地址。
- DWS集群的安全组规则需要确保DWS能接收来自客户端的网络访问。

  请确认DWS集群的安全组中存在符合如下要求的规则，如果不存在，请在DWS集群的安全组中添加相应的规则。
  - 方向：入方向
  - 协议：必须包含TCP，例如TCP、全部。
  - 端口：设置为DWS集群提供服务的数据库端口，例如“8000”。
  - 源地址：设置的IP地址需要包含DWS客户端主机的IP地址，例如“192.168.0.10/32”。

下载 Linux gsql 客户端并连接集群

步骤1 请参见**下载客户端**下载Linux gsql客户端，并使用SSH文件传输工具（例如WinSCP工具），将客户端工具上传到一个待安装Linux gsql的Linux主机上。

推荐下载使用与集群版本匹配的gsql工具，即8.1.0及以上版本集群使用8.1.x版本gsql、8.2.0及以上版本集群使用8.2.x版。若下载8.2.x版本gsql工具，需将dws_client_8.1.x_redhat_x64.zip替换为dws_client_8.2.x_redhat_x64.zip。此处仅以dws_client_8.1.x_redhat_x64.zip作为示例。

执行上传Linux gsql操作的用户需要对客户端主机的目标存放目录有完全控制权限。

步骤2 使用SSH会话工具，远程管理客户端主机。

弹性云服务器的登录方法请参见《弹性云服务器用户指南》中的“实例 > 登录Linux弹性云服务器 > SSH密码方式登录”章节。

步骤3 （可选）如果要使用SSL方式连接集群，请参考**使用SSL进行安全的TCP/IP连接**章节，在客户端主机配置SSL认证相关的参数。

说明

SSL连接方式的安全性高于非SSL方式，建议在客户端使用SSL连接方式。

步骤4 执行以下命令解压客户端工具。

```
cd <客户端存放路径>
unzip dws_client_8.1.x_redhat_x64.zip
```

其中：

- <客户端存放路径>：请替换为实际的客户端存放路径。
- dws_client_*8.1.x*_redhat_x64.zip：这是“RedHat x64”对应的客户端工具包名称，请替换为实际下载的包名。

步骤5 执行以下命令配置客户端。

```
source gsql_env.sh
```

提示以下信息表示客户端已配置成功

```
All things done.
```

步骤6 执行以下命令，使用gsql客户端连接DWS集群中的数据库。

```
gsql -d <数据库名称> -h <集群地址> -U <数据库用户> -p <数据库端口> -W <集群密码> -r
```

参数说明如下：

- “数据库名称”：输入所要连接的数据库名称。首次使用客户端连接集群时，请指定为集群的默认数据库“gaussdb”。
- “集群地址”：请参见《数据仓库服务用户指南》中"连接集群>获取集群连接地址"章节进行获取。如果通过公网地址连接，请指定为集群“公网访问地址”，如果通过内网地址连接，请指定为集群“内网访问地址”。
- “数据库用户”：输入集群数据库的用户名。首次使用客户端连接集群时，请指定为创建集群时设置的默认管理员用户，例如“dbadmin”。
- “数据库端口”：输入创建集群时设置的“数据库端口”。

例如，执行以下命令连接DWS集群的默认数据库gaussdb：

```
gsql -d gaussdb -h 10.168.0.74 -U dbadmin -p 8000 -W password -r
```

显示如下信息表示gsql工具已经连接成功：

```
gaussdb=>
```

----结束

gsql 命令参考

有关gsql的命令参考和更多信息，请参见**元命令参考**。

（可选）使用 gsql 导入 TPC-DS 样例数据

DWS支持用户将数据从集群外导入到集群中。用户可以参考以下指导，快速将样例数据从OBS导入集群，并对样例数据进行查询和分析。导入的样例数据是使用TPC-DS测试基准生成的标准性能测试数据。

TPC-DS是数据库决策支持测试基准。通过使用TPC-DS的测试数据以及测试案例，用户可以模拟真实场景下大数据集的统计、报表生成、联机查询、数据挖掘等复杂场景，从而了解数据库应用的功能和性能。

步骤1 使用SSH远程连接工具登录gsql客户端主机，并进入gsql目录，本例假设gsql客户端放在/opt目录下。

```
cd /opt
```

步骤2 执行以下命令，切换到指定目录并设置用户导入样例数据的用户密钥和OBS访问地址。

系统显示以下信息表示设置成功：

```
setup successfully!
```

说明

<Access_Key_Id>和<Secret_Access_Key>：分别表示访问密钥ID和私有访问密钥。请参见《数据仓库服务数据库开发指南》中的“导入数据 > 从OBS并行导入数据 > 创建访问密钥（AK和SK）”章节进行获取。然后将获取到的值替换到创建外表语句中。

步骤3 返回上一级目录，执行gsql环境变量。

```
cd ..
source gsql_env.sh
cd bin
```

步骤4 执行以下命令，将样例数据导入数据仓库。

命令格式：

```
gsql -d <数据库名称> -h <集群公网访问地址> -U <管理员用户> -p <数据仓库端口> -f <样例数据脚本保存路径> -r
```

命令示例：

```
gsql -d gaussdb -h 10.168.0.74 -U dbadmin -p 8000 -f /opt/sample/tpcds_load_data_from_obs.sql -r
```

说明

命令中样例数据脚本“tpcds_load_data_from_obs.sql”存放在DWS客户端的sample目录下，如“/opt/sample/”。

根据界面提示输入管理员密码，成功连接集群数据库后，系统会自动创建样例数据对应的外表用于关联集群外的数据，然后再创建存放样例数据的目标表，最后通过外表将数据导入到目标表中。

由于数据集较大，导入时间取决于当前DWS集群规格，一般为10~20分钟左右，等待系统显示如下执行时间信息表示导入成功，如下时间仅为示例。

```
Time:1845600.524 ms
```

步骤5 在Linux命令窗口，执行以下命令，切换到指定目录并查询样例数据。

```
cd /opt/sample/query_sql/
/bin/bash tpcds100x.sh
```

步骤6 根据命令提示，输入集群公网访问地址的IP地址、数据库端口、数据库名称、数据库访问用户以及用户密码。

- 数据库名称默认为“gaussdb”。
- 数据库访问用户和密码使用创建集群时配置的管理员用户和密码。

查询完成后，在当前查询目录，如“sample/query_sql/”下面会生成一个存放查询结果的目录，命名如“query_output_20170914_072341”。

----结束

3.3.2 使用 Windows gsql 客户端连接集群

用户在创建好数据仓库集群，开始使用集群数据库之前，需要使用数据库SQL客户端连接到数据库。DWS提供了与集群版本配套的Windows gsql命令行客户端工具，您可以使用Windows gsql客户端通过集群的公网地址或者内网地址访问集群。

操作步骤

步骤1 在计算机本地Windows操作系统服务器（Windows cmd）中安装和运行gsql客户端。Windows操作系统支持Windows Server 2008/Windows 7及以上。

步骤2 请参见下载客户端下载Windows gsql客户端，并将压缩包解压到本地文件夹中。

步骤3 在本地主机单击“开始”并搜索“cmd”，用管理员身份运行或单击快捷键“Win+R”打开Windows cmd窗口。

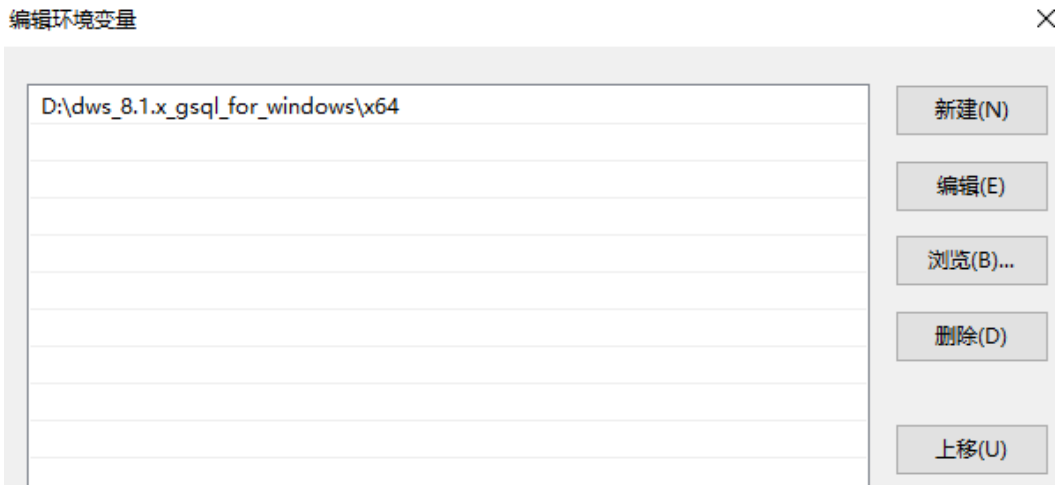
步骤4 设置环境变量，32位选择x86文件夹；64位选择x64文件夹。

方式一：命令行设置环境变量，打开Windows cmd窗口，执行set path=<window gsql>;%path%，其中<window gsql>为上一步骤解压Windows gsql客户端的文件夹路径。例如：

```
set path=C:\Users\xx\Desktop\dws_8.1.x_gsql_for_windows\x64;%path%
```

方式二：在控制面板中选择“系统->高级系统设置->高级->环境变量”，在系统环境变量Path中增加gsql路径。例如：

图 3-2 设置 Windows 环境变量



步骤5 （可选）如果要使用SSL方式连接集群，请参考**使用SSL进行安全的TCP/IP连接**章节，在客户端主机配置SSL认证相关的参数。

说明

SSL连接方式的安全性高于非SSL方式，建议在客户端使用SSL连接方式。

步骤6 在Windows cmd窗口执行以下命令，使用gsql客户端连接DWS集群中的数据库。

```
gsql -d <数据库名称> -h <集群地址> -U <数据库用户> -p <数据库端口> -W <集群密码> -r
```

参数说明如下：

- “数据库名称”：输入所要连接的数据库名称。首次使用客户端连接集群时，请指定为集群的默认数据库“gaussdb”。
- “集群地址”：请参见《数据仓库服务用户指南》中"连接集群>获取集群连接地址"章节进行获取。如果通过公网地址连接，请指定为集群“公网访问域名”，如果通过内网地址连接，请指定为集群“内网访问域名”。
- “数据库用户”：输入集群数据库的用户名。首次使用客户端连接集群时，请指定为创建集群时设置的默认管理员用户，例如“dbadmin”。
- “数据库端口”：输入创建集群时设置的“数据库端口”。

例如，执行以下命令连接DWS集群的默认数据库gaussdb：

```
gsql -d gaussdb -h 10.168.0.74 -U dbadmin -p 8000 -W password -r
```

显示如下信息表示gsql工具已经连接成功：

```
gaussdb=>
```

----结束

注意事项

1. Windows cmd默认的字符集是GBK，所以Windows gsql默认的client_encoding为GBK，部分UTF-8编码的字符无法在Windows gsql中显示。

   建议：-f执行的文件使用UTF-8编码，并设置默认的编码格式为UTF-8（set client_encoding=’utf-8’;）

2. Windows gsql中的路径需要使用‘/’作为分隔符，否则会报错。因为在元命令中‘\’是作为元命令开始的标志，在一般的单引号中，‘\’起转义作用。

   ```
   gaussdb=> \i D:\test.sql
   D:: Permission denied
   postgres=> \i D:/test.sql
   id
   ----
    1
   (1 row)
   ```

3. Windows gsql使用\!元命令执行系统命令时，需要使用系统命令要求的路径分隔符，一般是‘\’。

   ```
   gaussdb=> \! type D:/test.sql
   命令语法不正确。
   gaussdb=> \! type D:\test.sql
   select 1 as id;
   ```

4. Windows gsql不支持元命令\parallel。

   ```
   gaussdb=> \parallel
   ERROR: "\parallel" is not supported in Windows.
   ```

5. Linux shell中可以使用单引号和双引号作为字符串边界，但在Windows必须使用双引号作为字符串边界。

   ```
   gsql -h 192.168.233.189 -p 8109 -d postgres -U odbcuser -W password -c "select 1 as id"
    id
   ```

```
----
 1
(1 row)
```

使用单引号时报错，并忽略输入。

```
gsql -h 192.168.233.189 -p 8109 -d postgres -U odbcuser -W password -c 'select 1 as id'
gsql: warning: extra command-line argument "1" ignored
gsql: warning: extra command-line argument "as" ignored
gsql: warning: extra command-line argument "id'" ignored
ERROR:  unterminated quoted string at or near "'select"
LINE 1: 'select
```

6. Windows gsql在建立连接之后长时间未使用，连接session超时，会出现SSL报错，需要重新登录。报错如下：

   ```
   SSL SYSCALL error: Software caused connection abort (0x00002745/10053), remote datanode <NULL>, error: Result too large
   ```

7. Windows下Ctrl+C退出gsql。在当前行输入SQL语句时，若捕获到Ctrl+C信号后，无法将状态调整到重新输入的状态，会按照当前没有输入处理，将直接退出gsql。

   在输入as后执行Ctrl+C，输出\q后退出gsql。

   ```
   gaussdb=> select 1
   gaussdb=> as \q
   ```

8. Windows gsql不支持连接字符集为LATIN1的数据库，报错信息为：

   ```
   gsql: FATAL: conversion between GBK and LATIN1 is not supported
   ```

9. gsqlrc.conf文件的位置。

   默认的gsqlrc路径为%APPDATA%/postgresql/gsqlrc.conf，也可通过PSQLRC变量设置。

   ```
   set PSQLRC=C:\Users\xx\Desktop\dws_8.1.x_gsql_for_windows\x64\gsqlrc.conf
   ```

gsql 命令参考

有关gsql的命令参考和更多信息，请参见**元命令参考**。

3.3.3 使用 SSL 进行安全的 TCP/IP 连接

DWS支持SSL标准协议，SSL协议是安全性更高的协议标准，它们加入了数字签名和数字证书来实现客户端和服务器的双向身份验证，保证了通信双方更加安全的数据传输。为支持SSL连接方式，DWS已经从CA认证中心申请到正式的服务器、客户端的证书和密钥（假设服务器的私钥为server.key，证书为server.crt，客户端的私钥为client.key，证书为client.crt，CA根证书名称为cacert.pem）。

SSL连接方式的安全性高于普通模式，集群默认开启SSL功能允许来自客户端的SSL连接或非SSL连接，从安全性考虑，建议用户在客户端使用SSL连接方式。并且DWS服务器端的证书、私钥以及根证书已经默认配置完成。如果要强制使用SSL连接，需要在集群“安全设置”页面开启“服务器端是否强制使用SSL连接”，操作详情可参见**设置SSL连接**，客户端和服务器端SSL连接参数组合情况可请见**客户端和服务器端SSL连接参数组合情况**。

客户端或JDBC/ODBC应用程序使用SSL连接方式，用户必须在客户端或应用程序代码中配置相关的SSL连接参数。DWS管理控制台提供了客户端所需的SSL证书，该SSL证书包含了客户端所需的默认证书、私钥、根证书以及私钥密码加密文件。请将该SSL证书下载到客户端所在的主机上，然后在客户端中指定证书所在的路径，操作详情请参见**在gsql客户端配置SSL认证相关的数字证书参数**，SSL认证及客户端参数介绍可参见**SSL认证方式及客户端参数介绍**。

说明

使用默认的证书可能存在安全风险，为了提高系统安全性，强烈建议用户定期更换证书以避免被破解的风险。如果需要更换证书，请联系。

设置 SSL 连接

前提条件

- 修改安全配置参数并保存后，生效可能需要重启集群，否则将导致集群暂时不可用。
- 修改集群安全配置必须同时满足以下两个条件：
  - 集群状态为“可用”、“待重启”或“非均衡”。
  - 任务信息不能处于“创建快照中”、“节点扩容”、“配置中”或“重启中”。

操作步骤

步骤1 登录DWS控制台。

步骤2 在左侧导航树中，单击“专属集群 > 集群列表”。

步骤3 在集群列表中，单击指定集群的名称，然后单击“安全设置”。

默认显示“配置状态”为“已同步”，表示页面显示的是数据库当前最新结果。

步骤4 在“SSL连接”区域中，单击“服务器端是否强制使用SSL连接”的设置开关进行设置，建议开启。

 ：开启，设置参数**require_ssl**=1，表示服务器端强制要求SSL连接。

 ：关闭，设置参数**require_ssl**=0，表示服务器端对是否通过SSL连接不作强制要求，默认为关闭。设置require_ssl参数详情请参见**require_ssl（服务器）**。

说明

- 如果使用DWS提供的gsql客户端或ODBC驱动，DWS支持的SSL协议为TLSv1.2。
- 如果使用DWS提供的JDBC驱动，支持的SSL协议有SSLv3、TLSv1、TLSv1.1、TLSv1.2。客户端与数据库之间实际使用何种SSL协议，依赖客户端使用的JDK（Java Development Kit）版本，一般JDK支持多个SSL协议。

步骤5 单击“应用”。

系统将自动应用保存SSL连接设置，在“安全设置”页面，“配置状态”显示“应用中”。当“配置状态”显示为“已同步”，表示配置已保存生效。

----结束

在 gsql 客户端配置 SSL 认证相关的数字证书参数

DWS在集群部署完成后，默认已开启SSL认证模式。服务器端证书，私钥以及根证书已经默认配置完成。用户需要配置客户端的相关参数。

步骤1 登录DWS控制台，在左侧导航栏中，进入“连接客户端”页面。

步骤2 在“下载驱动程序”区域，单击“下载SSL证书”进行下载。

步骤3 使用文件传输工具（例如WinSCP工具）将SSL证书上传到客户端主机。

例如，将下载的证书“dws_ssl_cert.zip”存放到“/home/dbadmin/dws_ssl/”目录下。

步骤4 使用SSH远程连接工具（例如PuTTY）登录gsql客户端主机，然后执行以下命令进入SSL证书的存放目录，并解压SSL证书：

```
cd /home/dbadmin/dws_ssl/
unzip dws_ssl_cert.zip
```

步骤5 在gsql客户端主机上，执行export命令，配置SSL认证相关的数字证书参数。

SSL认证有两种认证方式：双向认证和单向认证。认证方式不同用户所需配置的客户端环境变量也不同，详细介绍请参见**SSL认证方式及客户端参数介绍**。

双向认证需配置如下参数：

```
export PGSSLCERT="/home/dbadmin/dws_ssl/sslcert/client.crt"
export PGSSLKEY="/home/dbadmin/dws_ssl/sslcert/client.key"
export PGSSLMODE="verify-ca"
export PGSSLROOTCERT="/home/dbadmin/dws_ssl/sslcert/cacert.pem"
```

单向认证需要配置如下参数：

```
export PGSSLMODE="verify-ca"
export PGSSLROOTCERT="/home/dbadmin/dws_ssl/sslcert/cacert.pem"
```

须知

- 从安全性考虑，建议使用双向认证方式。
- 配置客户端环境变量，必须包含文件的绝对路径。

步骤6 修改客户端密钥的权限。

客户端根证书、密钥、证书以及密钥密码加密文件需保证权限为600。如果权限不满足要求，则客户端无法以SSL方式连接到集群。

```
chmod 600 client.key
chmod 600 client.crt
chmod 600 client.key.cipher
chmod 600 client.key.rand
chmod 600 cacert.pem
```

----结束

SSL 认证方式及客户端参数介绍

SSL认证有两种认证方式，如**表3-8**所示。从安全性考虑，建议使用双向认证方式。

表 3-8 认证方式

认证方式	含义	配置客户端环境变量	维护建议
双向认证（推荐）	客户端验证服务器证书的有效性，同时服务器端也要验证客户端证书的有效性，只有认证成功，连接才能建立。	设置如下环境变量： • PGSSLCERT • PGSSLKEY • PGSSLROOTCERT • PGSSLMODE	该方式应用于安全性要求较高的场景。使用此方式时，建议设置客户端的PGSSLMODE变量为verify-ca。确保了网络数据的安全性。
单向认证	客户端只验证服务器证书的有效性，而服务器端不验证客户端证书的有效性。服务器加载证书信息并发送给客户端，客户端使用根证书来验证服务器端证书的有效性。	设置如下环境变量： • PGSSLROOTCERT • PGSSLMODE	为防止基于TCP链接的安全攻击，建议使用SSL证书认证功能。除配置客户端根证书外，建议客户端使用PGSSLMODE变量为verify-ca方式连接。

在客户端配置SSL认证相关的环境变量，详细信息请参见**表3-9**。

说明

客户端环境变量的路径以“*/home/dbadmin*/dws_ssl/”为例，在实际操作中请使用实际路径进行替换。

表 3-9 客户端参数

环境变量	描述	取值说明
PGSSLCERT	指定客户端证书文件，包含客户端的公钥。客户端证书用于表明客户端身份的合法性，公钥将发送给对端用来对数据进行加密。	必须包含文件的绝对路径，如： export PGSSLCERT='*/home/dbadmin/dws_ssl/sslcert/client.crt*' **默认值**：空
PGSSLKEY	指定客户端私钥文件，用于数字签名和对公钥加密的数据进行解密。	必须包含文件的绝对路径，如： export PGSSLKEY='*/home/dbadmin/dws_ssl/sslcert/client.key*' **默认值**：空

环境变量	描述	取值说明
PGSSLMODE	设置是否和服务器进行SSL连接协商，以及指定SSL连接的优先级。	**取值及含义：** ● disable：只尝试非SSL连接。 ● allow：首先尝试非SSL连接，如果连接失败，再尝试SSL连接。 ● prefer：首先尝试SSL连接，如果连接失败，将尝试非SSL连接。 ● require：只尝试SSL连接。如果存在CA文件，则按设置成verify-ca的方式验证。 ● verify-ca：只尝试SSL连接，并且验证服务器是否具有由可信任的证书机构签发的证书。 ● verify-full：DWS不支持此模式。 **默认值：**prefer **说明** 若集群外访问客户端时，部分节点出现报错：ssl SYSCALL error。则可执行export PGSSLMODE="allow"或export PGSSLMODE="prefer"。
PGSSLROOTCERT	指定为客户端颁发证书的根证书文件，根证书用于验证服务器证书的有效性。	必须包含文件的绝对路径，如： export PGSSLROOTCERT='*/home/dbadmin/dws_ssl/sslcert/certca.pem*' **默认值：**空
PGSSLCRL	指定证书吊销列表文件，用于验证服务器证书是否在废弃证书列表中，如果在，则服务器证书将会被视为无效证书。	必须包含文件的绝对路径，如： export PGSSLCRL='*/home/dbadmin/dws_ssl/sslcert/sslcrl-file.crl*' **默认值：**空

客户端和服务器端 SSL 连接参数组合情况

客户端最终是否使用SSL加密连接方式、是否验证服务器证书，取决于客户端参数sslmode与服务器端（即DWS集群侧）参数ssl、require_ssl。参数说明如下：

- **ssl（服务器）**

  ssl参数表示是否开启SSL功能。on表示开启，off表示关闭。

  - 对于集群版本高于1.3.1（包括1.3.1）的集群，默认为on，不支持在DWS管理控制台上设置。
  - 对于集群版本低于1.3.1的集群，默认为on。ssl参数可通过DWS管理控制台上集群的“安全设置”页面中的“SSL连接”进行设置。

- **require_ssl（服务器）**

  require_ssl参数是设置服务器端是否强制要求SSL连接，该参数只有当ssl为on时才有效。on表示服务器端强制要求SSL连接。off表示服务器端对是否通过SSL连接不作强制要求。

  - 对于集群版本高于1.3.1（包括1.3.1）的集群，默认为off。require_ssl参数可通过DWS管理控制台上集群的“安全设置”页面中的“服务器端是否强制使用SSL连接”进行设置。
  - 对于集群版本低于1.3.1的集群，默认为off，不支持在DWS管理控制台上设置。

- **sslmode（客户端）**

  可在SQL客户端工具中进行设置。

  - 在gsql命令行客户端中，为“PGSSLMODE”参数。
  - 在Data Studio客户端中，为“SSL模式”参数。

客户端参数sslmode与服务器端参数ssl、require_ssl配置组合结果如下：

表 3-10 客户端与服务器端 SSL 参数组合结果

ssl（服务器）	sslmode（客户端）	require_ssl（服务器）	结果
on	disable	on	由于服务器端要求使用SSL，但客户端针对该连接禁用了SSL，因此无法建立连接。
	disable	off	连接未加密。
	allow	on	连接经过加密。
	allow	off	连接未加密。
	prefer	on	连接经过加密。
	prefer	off	连接经过加密。
	require	on	连接经过加密。
	require	off	连接经过加密。
	verify-ca	on	连接经过加密，且验证了服务器证书。
	verify-ca	off	连接经过加密，且验证了服务器证书。
off	disable	on	连接未加密。
	disable	off	连接未加密。
	allow	on	连接未加密。
	allow	off	连接未加密。
	prefer	on	连接未加密。
	prefer	off	连接未加密。
	require	on	由于客户端要求使用SSL，但服务器端禁用了SSL，因此无法建立连接。
	require	off	由于客户端要求使用SSL，但服务器端禁用了SSL，因此无法建立连接。

ssl（服务器）	sslmode（客户端）	require_ssl（服务器）	结果
	verify-ca	on	由于客户端要求使用SSL，但服务器端禁用了SSL，因此无法建立连接。
	verify-ca	off	由于客户端要求使用SSL，但服务器端禁用了SSL，因此无法建立连接。

3.4 获取帮助

操作步骤

- 连接数据库时，可以使用如下命令获取帮助信息。

  ```
  gsql --help
  ```

  显示如下帮助信息：

  ```
  ......
  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
  ......
  ```

- 连接到数据库后，可以使用如下命令获取帮助信息。

  ```
  help
  ```

  显示如下帮助信息：

  ```
  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
  ```

任务示例

步骤1 查看gsql的帮助信息。具体执行命令请参见表3-11。

表 3-11 使用 gsql 联机帮助

描述	示例
查看版权信息	\copyright

描述	示例
查看DWS支持的SQL语句的帮助	查看DWS支持的SQL语句的帮助 例如，查看DWS支持的所有SQL语句： \h Available help: ABORT ALTER DATABASE ALTER DATA SOURCE ... ... 例如，查看CREATE DATABASE命令的参数可使用下面的命令： \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]}[...]];
查看gsql命令的帮助	例如，查看gsql支持的命令： \? General \copyright show PostgreSQL usage and distribution terms \g [FILE] or ; execute query (and send results to file or |pipe) \h(\help) [NAME] help on syntax of SQL commands, * for all commands \q quit gsql ... ...

----结束

3.5 命令参考

详细的gsql参数请参见**表3-12**、**表3-13**、**表3-14**和**表3-15**。

表 3-12 常用参数

参数	参数说明	取值范围
-c, --command=COMMAND	声明gsql要执行一条字符串命令然后退出。	-
-C, --set-file=FILENAME	使用文件作为命令源而不是交互式输入，gsql处理完文件后不退出，继续处理其他内容。	绝对路径或相对路径，且满足操作系统路径命名规则。
-d, --dbname=DBNAME	指定想要连接的数据库名称。	字符串。

参数	参数说明	取值范围
-D, --dynamic-param	用于控制执行SQL语句设置变量和${}变量引用方式，具体示例参见**变量**。	-
-f, --file=FILENAME	使用文件作为命令源而不是交互式输入。gsql将在处理完文件后结束。如果FILENAME是-（连字符），则从标准输入读取。	绝对路径或相对路径，且满足操作系统路径命名规则。
-l, --list	列出所有可用的数据库，然后退出。	-
-v, --set, --variable=NAME=VALUE	设置gsql变量NAME为VALUE。 变量的示例和详细说明请参见**变量**。	-
-X, --no-gsqlrc	不读取启动文件（系统范围的gsqlrc或者用户的~/.gsqlrc都不读取）。 **说明** 启动文件默认为~/.gsqlrc，或通过PSQLRC环境变量指定。	-
-1 ("one"), --single-transaction	当gsql使用-f选项执行脚本时，会在脚本的开头和结尾分别加上START TRANSACTION/COMMIT用于把整个脚本当作一个事务执行。这将保证该脚本完全执行成功，或者脚本无效。 **说明** 如果脚本中已经使用了START TRANSACTION，COMMIT，ROLLBACK，则该选项无效。	-
-?, --help	显示关于gsql命令行参数的帮助信息然后退出。	-
-V, --version	打印gsql版本信息然后退出。	-

表 3-13 输入和输出参数

参数	参数说明	取值范围
-a, --echo-all	在读取行时向标准输出打印所有内容。 **注意** 使用此参数可能会暴露部分SQL语句中的敏感信息，如创建用户语句中的password信息等，请谨慎使用。	-
-e, --echo-queries	把所有发送给服务器的查询同时回显到标准输出。 **注意** 使用此参数可能会暴露部分SQL语句中的敏感信息，如创建用户语句中的password信息等，请谨慎使用。	-
-E, --echo-hidden	回显由\d和其他反斜杠命令生成的实际查询。	-

参数	参数说明	取值范围
-k, --with-key=KEY	使用gsql对导入的加密文件进行解密。 **须知** 对于本身就是shell命令中的关键字符如单引号（'）或双引号（"），Linux shell会检测输入的单引号（'）或双引号（"）是否匹配。如果不匹配，shell认为用户没有输入完毕，会一直等待用户输入，从而不会进入到gsql程序。	-
-L, --log-file=FILENAME	除了正常的输出源之外，把所有查询输出记录到文件FILENAME中。 **注意** • 使用此参数可能会暴露部分SQL语句中的敏感信息，如创建用户语句中的password信息等，请谨慎使用。 • 此参数只保留查询结果到相应文件中，主要目标是为了查询结果能够更好更准确地被其他调用者（例如自动化运维脚本）解析；而不是保留gsql运行过程中的相关日志信息。	绝对路径或相对路径，且满足操作系统路径命名规则。
-m, --maintenance	允许在两阶段事务恢复期间连接集群。 **说明** 该选项是一个开发选项，禁止用户使用，只限专业技术人员使用，功能是：使用该选项时，gsql可以连接到备机，用于校验主备机数据的一致性。	-
-n, --no-libedit	关闭命令行编辑。	-
-o, --output=FILENAME	将所有查询输出重定向到文件FILENAME。	绝对路径或相对路径，且满足操作系统路径命名规则。
-q, --quiet	安静模式，执行时不会打印出额外信息。	缺省时gsql将打印许多其他输出信息。
-s, --single-step	单步模式运行。意味着每个查询在发往服务器之前都要提示用户，用这个选项也可以取消执行。此选项主要用于调试脚本。 **注意** 使用此参数可能会暴露部分SQL语句中的敏感信息，如创建用户语句中的password信息等，请谨慎使用。	-
-S, --single-line	单行运行模式，这时每个命令都将由换行符结束，像分号那样。	-

表 3-14 输出格式参数

参数	参数说明	取值范围
-A, --no-align	切换为非对齐输出模式。	缺省为对齐输出模式。

参数	参数说明	取值范围
-F, --field-separator=STRING	设置域分隔符（默认为“|”）。	-
-H, --html	打开HTML格式输出。	-
-P, --pset=VAR[=ARG]	在命令行上以\pset的风格设置打印选项。 **说明** 这里必须用等号而不是空格分隔名称和值。例如，把输出格式设置为LaTeX，可以键入-P format=latex	-
-R, --record-separator=STRING	设置记录分隔符。	-
-r	开启客户端操作历史记录功能。	缺省为关闭。
-t, --tuples-only	只打印行。	-
-T, --table-attr=TEXT	允许声明放在HTML table标签里的选项。 使用时请搭配参数“-H,--html”，指定为HTML格式输出。	-
-x, --expanded	打开扩展表格式模式。	-
-z, --field-separator-zero	设置非对齐输出模式的域分隔符为空。 使用时请搭配参数“-A, --no-align”，指定为非对齐输出模式。	-
-0, --record-separator-zero	设置非对齐输出模式的记录分隔符为空。 使用时请搭配参数“-A, --no-align”，指定为非对齐输出模式。	-
-g	显示所有SQL语句和指定文件的分隔符。 **说明** -g参数必须和-f参数一起设置。	-

表 3-15 连接参数

参数	参数说明	取值范围
-h, --host=HOSTNAME	指定正在运行服务器的主机名或者Unix域套接字的路径。	如果省略主机名，gsql将通过Unix域套接字与本地主机的服务器相连，或者在没有Unix域套接字的机器上，通过TCP/IP与localhost连接。
-p, --port=PORT	指定数据库服务器的端口号。 可以通过port参数修改默认端口号。	默认为8000。
-U, --username=USERNAME	指定连接数据库的用户。 **说明** • 通过该参数指定用户连接数据库时，需要同时提供用户密码用于身份验证。您可以通过交换方式输入密码，或者通过-W参数指定密码。 • 用户名中包含有字符$，需要在字符$前增加转义字符才可成功连接数据库。	字符串。默认使用与当前操作系统用户同名的用户。
-W, --password=PASSWORD	当使用-U参数连接远端数据库时，可通过该选项指定密码。 **说明** 用户密码中包含特殊字符“\”和"`"时，需要增加转义字符才可成功连接数据库。 如果用户未输入该参数，但是数据库连接需要用户密码，这时将出现交互式输入，请用户输入当前连接的密码。该密码最大长度为999字节，受限于GUC参数password max length的最大值。	符合密码复杂度要求。

3.6 元命令参考

介绍使用DWS数据库命令行交互工具登录数据库后，gsql所提供的元命令。所谓元命令就是在gsql里输入的任何以不带引号的反斜杠开头的命令。

注意事项

- 一个gsql元命令的格式是反斜杠后面紧跟一个动词，然后是任意参数。参数命令动词和其他参数以任意个空白字符间隔。
- 要在参数里面包含空白，必须用单引号把它引起来。要在这样的参数里包含单引号，可以在前面加一个反斜杠。任何包含在单引号里的内容都会被进一步进行类似C语言的替换：\n（新行）、\t（制表符）、\b（退格）、\r（回车）、\f（换页）、\digits（八进制表示的字符）、\xdigits（十六进制表示的字符）。
- 用""包围的内容被当做一个命令行传入shell。该命令的输出（删除了结尾的新行）被当做参数值。

- 如果不带引号的参数以冒号（:）开头，它会被当做一个gsql变量，并且该变量的值最终会成为真正的参数值。
- 有些命令以一个SQL标识的名称（比如一个表）为参数。这些参数遵循SQL语法关于双引号的规则：不带双引号的标识强制转换成小写，而双引号保护字母不进行大小写转换，并且允许在标识符中使用空白。在双引号中，成对的双引号在结果名字中分析成一个双引号。比如，FOO"BAR"BAZ解析成fooBARbaz；而"A weird"" name"解析成A weird" name。
- 对参数的分析在遇到另一个不带引号的反斜杠时停止。这里会认为是一个新的元命令的开始。特殊的双反斜杠序列（\\）标识参数的结尾并将继续分析后面的SQL语句（如果存在）。这样SQL和gsql命令可以自由地在一行里面混合。但是在任何情况下，一条元命令的参数不能延续超过行尾。

元命令

元命令的详细说明请参见表3-16、表3-17、表3-18、表3-19、表3-21、表3-23、表3-24、表3-25和表3-27。

须知

以下命令中所提到的FILE代表文件路径。此路径可以是绝对路径（如/home/gauss/file.txt），也可以是相对路径（file.txt，file.txt会默认在用户执行gsql命令所在的路径下创建）。

表 3-16 一般的元命令

参数	参数说明	取值范围
\copyright	显示DWS的版本和版权信息。	-
\g [FILE] or ;	执行查询（并将结果发送到文件或管道）。	-
\h(\help) [NAME]	给出指定SQL语句的语法帮助。	如果没有给出NAME，gsql将列出可获得帮助的所有命令。如果NAME是一个星号（*），则显示所有SQL语句的语法帮助。

参数	参数说明	取值范围
\parallel [on [num]|off]	控制并发执行开关。 ● on：打开控制并发执行开关，且最大并发数为num。 ● off：关闭控制并发执行开关。 **说明** ● 不支持事务中开启并发执行以及并发中开启事务。 ● 不支持\d这类元命令的并发。 ● 并发select返回结果混乱问题，此为客户可接受，core、进程停止响应不可接受。 ● 不推荐在并发中使用set语句，否则导致结果与预期不一致。 ● 不支持在\parallel中使用DISCARD命令。 ● 不支持创建临时表！如需使用临时表，需要在开启parallel之前创建好，并在parallel内部使用。parallel内部不允许创建临时表。 ● \parallel执行时最多会启动num个独立的gsql进程连接服务器。 ● \parallel中所有作业的持续时间不能超过session_timeout，否则可能会导致并发执行过程中断连。 ● \parallel不支持对VOLATILE/GLOBAL全局临时表进行操作。	num的默认值：1024。 **须知** ● 服务器能接受的最大连接数受max_connection及当前已有连接数限制。 ● 设置num时请考虑服务器当前可接受的实际连接数合理指定。
\q [value]	退出gsql程序。在一个脚本文件里，只在脚本终止的时候执行。退出码可由value值决定。	-

表 3-17 查询缓存区元命令

参数	参数说明
\e [FILE] [LINE]	使用外部编辑器编辑查询缓冲区（或者文件）。
\ef [FUNCNAME [LINE]]	使用外部编辑器编辑函数定义。如果指定了LINE（即行号），则光标会指到函数体的指定行。
\p	打印当前查询缓冲区到标准输出。
\r	重置（或清空）查询缓冲区。
\w FILE	将当前查询缓冲区输出到文件。

表 3-18 输入/输出元命令

参数	参数说明
\copy { table [(column_list)] | (query) } { from | to } { filename | stdin | stdout | pstdin | pstdout } [with] [binary] [oids] [delimiter [as] 'character'] [null [as] 'string'] [csv [header] [quote [as] 'character'] [escape [as] 'character'] [force quote column_list | *] [force not null column_list]]	在任何gsql客户端登录数据库成功后可以执行导入导出数据， 这是一个运行SQL COPY命令的操作，但不是读取或写入指定文件的服务器，而是读取或写入文件，并在服务器和本地文件系统之间路由数据。 这意味着文件的可访问性和权限是本地用户的权限，而不是服务器的权限，并且不需要数据库初始化用户权限。 **说明** \COPY只适合小批量，格式良好的数据导入，容错能力较差。导入数据应优先选择GDS或COPY。
\echo [STRING]	把字符串写到标准输出。
\i FILE	从文件FILE中读取内容，并将其当作输入，执行查询。
\i+ FILE KEY	执行加密文件中的命令。
\ir FILE	和\i类似，只是相对于存放当前脚本的路径。
\ir+ FILE KEY	和\i+类似，只是相对于存放当前脚本的路径。
\o [FILE]	把所有的查询结果发送到文件里。
\qecho [STRING]	把字符串写到查询结果输出流里。

说明

表3-19中的选项S表示显示系统对象，+表示显示对象附加的描述信息。PATTERN用来指定要被显示的对象名称。

表 3-19 显示信息元命令

参数	参数说明	取值范围	示例
\d[S+]	列出当前search_path中模式下所有的表、视图和序列。当search_path中不同模式存在同名对象时，只显示search_path中位置靠前模式下的同名对象。	-	列出当前search_path中模式下所有的表、视图和序列。 `\d`
\d[S+] NAME	列出指定表结构、视图结构和索引的结构。	-	假设存在表a，列出指定表a的表结构。 `\dtable+ a`

参数	参数说明	取值范围	示例
\d+ [PATTERN]	列出所有表、视图和索引。	如果声明了PATTERN，只显示名字匹配PATTERN的表、视图和索引。	列出所有名称以f开头的表、视图和索引。 \d+ f*
\da[S] [PATTERN]	列出所有可用的聚集函数，以及它们操作的数据类型和返回值类型。	如果声明了PATTERN，只显示名字匹配PATTERN的聚集函数。	列出所有名称以f开头可用的聚集函数，以及它们操作的数据类型和返回值类型。 \da f*
\db[+] [PATTERN]	列出所有可用的表空间。	如果声明了PATTERN，只显示名字匹配PATTERN的表空间。	列出所有名称以p开头的可用表空间。 \db p*
\dc[S+] [PATTERN]	列出所有字符集之间的可用转换。	如果声明了PATTERN，只显示名字匹配PATTERN的转换。	列出所有字符集之间的可用转换。 \dc *
\dC[+] [PATTERN]	列出所有类型转换。	如果声明了PATTERN，只显示名字匹配PATTERN的转换。	列出所有名称以c开头的类型转换。 \dC c*
\dd[S] [PATTERN]	显示所有匹配PATTERN的描述。	如果没有给出参数，则显示所有可视对象。“对象”包括：聚集、函数、操作符、类型、关系(表、视图、索引、序列、大对象)、规则。	列出所有可视对象。 \dd
\ddp [PATTERN]	显示所有默认的使用权限。	如果指定了PATTERN，只显示名字匹配PATTERN的使用权限。	列出所有默认的使用权限。 \ddp
\dD[S+] [PATTERN]	列出所有可用域。	如果声明了PATTERN，只显示名字匹配PATTERN的域。	列出所有可用域。 \dD
\ded[+] [PATTERN]	列出所有的Data Source对象。	如果声明了PATTERN，只显示名字匹配PATTERN的对象。	列出所有的Data Source对象。 \ded

参数	参数说明	取值范围	示例
\det[+] [PATTERN]	列出所有的外部表。	如果声明了PATTERN，只显示名字匹配PATTERN的表。	列出所有的外部表。 \det
\des[+] [PATTERN]	列出所有的外部服务器。	如果声明了PATTERN，只显示名字匹配PATTERN的服务器。	列出所有的外部服务器。 \des
\deu[+] [PATTERN]	列出用户映射信息。	如果声明了PATTERN，只显示名字匹配PATTERN的信息。	列出用户映射信息。 \deu
\dew[+] [PATTERN]	列出封装的外部数据。	如果声明了PATTERN，只显示名字匹配PATTERN的数据。	列出封装的外部数据。 \dew
\df[antw][S+] [PATTERN]	列出所有可用函数，以及它们的参数和返回的数据类型。a代表聚集函数，n代表普通函数，t代表触发器，w代表窗口函数。	如果声明了PATTERN，只显示名字匹配PATTERN的函数。	列出所有可用函数，以及它们的参数和返回的数据类型。 \df
\dF[+] [PATTERN]	列出所有的文本搜索配置信息。	如果声明了PATTERN，只显示名字匹配PATTERN的配置信息。	列出所有的文本搜索配置信息。 \dF+
\dFd[+] [PATTERN]	列出所有的文本搜索字典。	如果声明了PATTERN，只显示名字匹配PATTERN的字典。	列出所有的文本搜索字典。 \dFd
\dFp[+] [PATTERN]	列出所有的文本搜索分析器。	如果声明了PATTERN，只显示名字匹配PATTERN的分析器。	列出所有的文本搜索分析器。 \dFp
\dFt[+] [PATTERN]	列出所有的文本搜索模板。	如果声明了PATTERN，只显示名字匹配PATTERN的模板。	列出所有的文本搜索模板。 \dFt
\dg[+] [PATTERN]	列出所有数据库角色。 **说明** 因为用户和群组的概念被统一为角色，所以这个命令等价于\du。为了和以前兼容，所以保留两个命令。	如果指定了PATTERN，只显示名字匹配PATTERN的角色。	列出名字为‘j_e’所有数据库角色。 \dg j?e

参数	参数说明	取值范围	示例
\dl	\lo_list的别名，显示一个大对象的列表。	-	列出所有的大对象。 \dl
\dL[S+] [PATTERN]	列出可用的程序语言。	如果指定了PATTERN，只列出名字匹配PATTERN的语言。	列出可用的程序语言。 \dL
\dn[S+] [PATTERN]	列出所有的模式（名字空间）。	如果声明了PATTERN，只列出名字匹配PATTERN的模式名。缺省时，只列出用户创建的模式。	列出所有名称以d开头的模式以及相关信息。 \dn+ d*
\do[S] [PATTERN]	列出所有可用的操作符，以及它们的操作数和返回的数据类型。	如果声明了PATTERN，只列出名字匹配PATTERN的操作符。缺省时，只列出用户创建的操作符。	列出所有可用的操作符，以及它们的操作数和返回的数据类型。 \do
\dO[S+] [PATTERN]	列出排序规则。	如果声明了PATTERN，只列出名字匹配PATTERN的规则。缺省时，只列出用户创建的规则。	列出排序规则。 \dO
\dp [PATTERN]	列出一列可用的表、视图以及相关的权限信息。 \dp显示结果如下： rolename=xxxx/yyyy --赋予一个角色的权限 =xxxx/yyyy --赋予public的权限 xxxx表示赋予的权限，yyyy表示授予这个权限的角色。权限的参数说明请参见**表3-20**。	如果指定了PATTERN，只列出名字匹配PATTERN的表、视图。	列出一列可用的表、视图以及相关的权限信息。 \dp
\drds [PATTERN1 [PATTERN2]]	列出所有修改过的配置参数。这些设置可以是针对角色的、针对数据库的或者同时针对两者的。PATTERN1和PATTERN2表示要列出的角色PATTERN和数据库PATTERN。	如果声明了PATTERN，只列出名字匹配PATTERN的规则。缺省或指定*时，则会列出所有设置。	列出数据库所有修改过的配置参数。 \drds *
\dRp[+] [PATTERN]	列出所有的发布。该元命令仅8.2.0.100及以上集群版本支持。	如果指定了PATTERN，只列出名字匹配PATTERN的发布。	列出所有的发布。 \dRp

参数	参数说明	取值范围	示例
\dRs[+] [PATTERN]	列出所有的订阅。该元命令仅8.2.0.100及以上集群版本支持。	如果指定了PATTERN，只列出名字匹配PATTERN的订阅。	列出所有的订阅。 \dRs
\dT[S+] [PATTERN]	列出所有的数据类型。	如果指定了PATTERN，只列出名字匹配PATTERN的类型。	列出所有的数据类型。 \dT
\du[+] [PATTERN]	列出所有数据库角色。 **说明** 因为用户和群组的概念被统一为角色，所以这个命令等价于\dg。为了和以前兼容，所以保留两个命令。	如果指定了PATTERN，则只列出名字匹配PATTERN的角色。	列出所有数据库角色。 \du
\dE[S+] [PATTERN] \di[S+] [PATTERN] \ds[S+] [PATTERN] \dt[S+] [PATTERN] \dv[S+] [PATTERN]	这一组命令，字母E，i，s，t和v分别代表着外部表，索引，序列，表和视图。可以以任意顺序指定其中一个或者它们的组合来列出这些对象。例如：\dit列出所有的索引和表。在命令名称后面追加+，则每一个对象的物理尺寸以及相关的描述也会被列出。 **说明** 本版本暂时不支持序列。	如果指定了PATTERN，只列出名称匹配该PATTERN的对象。默认情况下只会显示用户创建的对象。通过PATTERN或者S修饰符可以把系统对象包括在内。	列出所有的索引和视图。 \div
\dx[+] [PATTERN]	列出安装数据库的扩展信息。	如果指定了PATTERN，则只列出名字匹配PATTERN的扩展信息。	列出安装数据库的扩展信息。 \dx
\l[+]	列出服务器上所有数据库的名字、所有者、字符集编码以及使用权限。	-	列出服务器上所有数据库的名字、所有者、字符集编码以及使用权限。 \l

参数	参数说明	取值范围	示例

\sf[+] FUNCNAME	显示函数的定义。
	说明
	对于带圆括号的函数名，需要在函数名两端添加双引号，并且在双引号后面加上参数类型列表。参数类型列表两端添加圆括号。
		-	假设存在函数function_a和函数名带圆括号的函数func()name，列出函数的定义。
			`\sf function_a`
			`\sf "func()name"(argtype1, argtype2)`

\z [PATTERN]	列出数据库中所有表、视图和序列，以及它们相关的访问特权。	如果给出任何pattern ，则被当成一个正则表达式，只显示匹配的表、视图、序列。	列出数据库中所有表、视图和序列，以及它们相关的访问特权。 `\z`

表 3-20 权限的参数说明

参数	参数说明
r	SELECT：允许对指定的表、视图读取数据。
w	UPDATE：允许对指定表更新字段。
a	INSERT：允许对指定表插入数据。
d	DELETE：允许删除指定表中的数据。
D	TRUNCATE：允许清理指定表中的数据。
x	REFERENCES：允许创建外键约束。
t	TRIGGER：允许在指定表上创建触发器。
X	EXECUTE：允许使用指定的函数，以及利用这些函数实现的操作符。
U	USAGE： ● 对于过程语言，允许用户在创建函数时，指定过程语言。 ● 对于模式，允许访问包含在指定模式中的对象。 ● 对于序列，允许使用nextval函数。
C	CREATE： ● 对于数据库，允许在该数据库里创建新的模式。 ● 对于模式，允许在该模式中创建新的对象。 ● 对于表空间，允许在其中创建表，以及允许创建数据库和模式的时候把该表空间指定为其缺省表空间。
c	CONNECT：允许用户连接到指定的数据库。

参数	参数说明
T	TEMPORARY：允许创建临时表。
A	ANALYZE|ANALYSE：允许分析表。
L	ALTER：允许修改表、模式或函数。
P	DROP：允许删除表、模式或函数。
v	VACUUM：允许对表执行VACUUM。
arwdDxtA, vLP	ALL PRIVILEGES：一次性给指定用户/角色赋予所有可赋予的权限。 vLP权限组为8.1.3及以上集群版本中新增表级权限ALTER/DROP/VACUUM，新增schema权限ALTER/DROP。
*	给前面权限的授权选项。

表 3-21 格式化元命令

参数	参数说明
\a	对齐模式和非对齐模式之间的切换。
\C [STRING]	把正在打印的表的标题设置为一个查询的结果或者取消这样的设置。
\f [STRING]	对于不对齐的查询输出，显示或者设置域分隔符。
\H	• 若当前模式为文本格式，则切换为HTML输出格式。 • 若当前模式为HTML格式，则切换回文本格式。
\pset NAME [VALUE]	设置影响查询结果表输出的选项。NAME的取值见**表3-22**。
\t [on|off]	切换输出的字段名的信息和行计数脚注。
\T [STRING]	指定在使用HTML输出格式时放在table标签里的属性。如果参数为空，不设置。
\x [on|off|auto]	切换扩展行格式。

表 3-22 可调节的打印选项

选项	选项说明	取值范围
border	value必须是一个数字。通常这个数字越大，表的边界就越宽线就越多，但是这个取决于特定的格式。	● 在HTML格式下，取值范围为大于0的整数。 ● 在其他格式下，取值范围： – 0：无边框 – 1：内部分隔线 – 2：台架
expanded (或 x)	在正常和扩展格式之间切换。	● 当打开扩展格式时，查询结果用两列显示，字段名称在左、数据在右。这个模式在数据无法放进通常的"水平"模式的屏幕时很有用。 ● 在正常格式下，当查询输出的格式比屏幕宽时，用扩展格式。正常格式只对aligned和wrapped格式有用。
fieldsep	声明域分隔符来实现非对齐输出。这样就可以创建其他程序希望的制表符或逗号分隔的输出。要设置制表符域分隔符，键入\pset fieldsep '\t'。缺省域分隔符是 '|' (竖条符)。	-
fieldsep_zero	声明域分隔符来实现非对齐输出到零字节。	-
footer	用来切换脚注。	-

选项	选项说明	取值范围
format	设置输出格式。允许使用唯一缩写（这意味着一个字母就够了）。	取值范围： ● unaligned：写一行的所有列在一条直线上中，当前活动字段分隔符分隔。 ● aligned：此格式是标准的，可读性最好的文本输出。 ● wrapped：类似aligned，但是包装跨行的宽数据值，使其适应目标字段的宽度输出。 ● html：把表输出为可用于文档里的对应标记语言。输出不是完整的文档。 ● latex：把表输出为可用于文档里的对应标记语言。输出不是完整的文档。 ● troff-ms：把表输出为可用于文档里的对应标记语言。输出不是完整的文档。
null	打印一个字符串，用来代替一个null值。	缺省是什么都不打印，这样很容易和空字符串混淆。
numericlocale	切换分隔小数点左边的数值的区域相关的分组符号。	● on：显示指定的分隔符。 ● off：不显示分隔符。 忽略此参数，显示默认的分隔符。
pager	控制查询和gsql帮助输出的分页器。如果设置了环境变量PAGER，输出将被指向到指定程序，否则使用系统缺省。	● on：当输出到终端且不适合屏幕显示时，使用分页器。 ● off：不使用分页器。 ● always：当输出到终端无论是否符合屏幕显示时，都使用分页器。
recordsep	声明在非对齐输出格式时的记录分隔符。	-
recordsep_zero	声明在非对齐输出到零字节时的记录分隔符。	-
tableattr（或T）	声明放在html输出格式中HTML table标签的属性（例如：cellpadding或bgcolor）。注意：这里可能不需要声明border，因为已经在\pset border里用过了。如果没有给出value，则不设置表的属性。	-

选项	选项说明	取值范围
title	为随后打印的表设置标题。这个可以用于给输出一个描述性标签。如果没有给出value，不设置标题。	-
tuples_only（或者t）	在完全显示和只显示实际的表数据之间切换。完全显示将输出像列头、标题、各种脚注等信息。在tuples_only模式下，只显示实际的表数据。	-

表 3-23 连接元命令

参数	参数说明	取值范围
\[connect] [DBNAME|- USER|- HOST|- PORT|-]	连接到一个新的数据库（当前数据库为gaussdb）。当数据库名称长度超过63个字节时，默认前63个字节有效，连接到前63个字节对应的数据库，但是gsql的命令提示符中显示的数据库对象名仍为截断前的名称。 **说明** 重新建立连接时，如果切换数据库登录用户，将可能会出现交互式输入，要求输入新用户的连接密码。该密码最大长度为999字节，受限于GUC参数password max length的最大值。	-
\encoding [ENCODING]	设置客户端字符编码格式。	不带参数时，显示当前的编码格式。
\conninfo	输出当前连接的数据库的信息。	-

表 3-24 操作系统元命令

参数	参数说明	取值范围
\cd [DIR]	切换当前的工作目录。	绝对路径或相对路径，且满足操作系统路径命名规则。
\setenv NAME [VALUE]	设置环境变量NAME为VALUE，如果没有给出VALUE值，则不设置环境变量。	-
\timing [on|off]	以毫秒为单位显示每条SQL语句的执行时间。	• on表示打开显示。 • off表示关闭显示。
\! [COMMAND]	返回到一个单独的Unix shell或者执行Unix命令COMMAND。	-

表 3-25 变量元命令

参数	参数说明
\prompt [TEXT] NAME	提示用户用文本格式来指定变量名字。
\set [NAME [VALUE]]	设置内部变量NAME为VALUE或者如果给出了多于一个值，设置为所有这些值的连接结果。如果没有给出第二个参数，只设变量不设值。 有一些常用变量被gsql特殊对待，它们是一些选项设置，通常所有特殊对待的变量都是由大写字母组成(可能还有数字和下划线)。 表3-26是一个所有特殊对待的变量列表。
\set-multi NAME [VALUE] \end-multi	设置内部变量NAME为VALUE，VALUE可以由多行字符串组成。\set-multi使用时，第二个参数必须给出。可参考表下面的\set-multi元命令使用示例。 **说明** \set-multi 和\end-multi中出现的元命令会被忽略。
\unset NAME	不设置（或删除）gsql变量名。

\set-multi元命令使用示例

示例文件test.sql：

```
\set-multi multi_line_var
select
    id,name
from
    student;
\end-multi
\echo multi_line_var is "${multi_line_var}"
\echo -------------------------
\echo result is
${multi_line_var}
```

gsql -d gaussdb -p 25308 --dynamic-param -f test.sql 执行结果：

```
multi_line_var is "select
        id,name
from
        student; "
-------------------------
result is
 id | name
----+-------
  1 | Jack
  2 | Tom
  3 | Jerry
  4 | Danny
(4 rows)
```

通过\set-multi \end-multi设置变量multi_line_var为一个SQL语句，并在后面通过动态变量解析获得这个变量。

示例文件test.sql：

```
\set-multi multi_line_var
select 1 as id;
select 2 as id;
```

```
\end-multi
\echo multi_line_var is "${multi_line_var}"
\echo -------------------------
\echo result is
${multi_line_var}
```

gsql -d -p 25308 --dynamic-param -f test.sql 执行结果：

```
multi_line_var is "select 1 as id;
select 2 as id;"
-------------------------
result is
 id
----
  1
(1 row)

 id
----
  2
(1 row)
```

通过\set-multi \end-multi设置变量multi_line_var为两个SQL语句，并在后面通过动态变量解析获得这个变量。因为变量中的内容以“;”结尾，gsql发送SQL语句并获得打印执行结果。

表 3-26 \set 常用命令

名称	命令说明	取值范围
\set VERBOSITY value	这个选项可以设置为值default， verbose，terse之一以控制错误报告的冗余行。	value取值范围：default，verbose，terse
\set ON_ERROR_STOP value	如果设置了这个变量，脚本处理将马上停止。如果该脚本是从另外一个脚本调用的，那个脚本也会按同样的方式停止。如果最外层的脚本不是从一次交互的gsql会话中调用的而是用-f选项调用的，gsql将返回错误代码3，以示这个情况与致命错误条件的区别(错误代码为1)。	value取值范围为：on/off，true/false，yes/no，1/0

名称	命令说明	取值范围
\set RETRY [retry_times]	用于控制是否开启语句出错场景下的重试功能，参数retry_times用来指定最大重试次数，缺省值为5，取值范围为5-10。当重试功能已经开启时，再次执行\set RETRY可以关闭该功能。 使用配置文件retry_errcodes.conf列举需要重试的错误码列表，该文件和gsql可执行程序位于同一级目录下。该配置文件为系统配置，非用户定义，不允许用户直接修改。 当前支持以下13类出错场景的重试： ● YY001：TCP通信错误，Connection reset by peer（ CN和DN间通信 ） ● YY002：TCP通信错误，Connection reset by peer（ DN和DN间通信 ） ● YY003：锁超时，Lock wait timeout.../ wait transaction xxx sync time exceed xxx ● YY004：TCP通信错误，Connection timed out ● YY005：SET命令发送失败，ERROR SET query ● YY006：内存申请失败，memory is temporarily unavailable ● YY007：通信库错误，Memory allocate error ● YY008：通信库错误，No data in buffer ● YY009：通信库错误，Close because release memory ● YY010：通信库错误，TCP disconnect ● YY011：通信库错误，SCTP disconnect ● YY012：通信库错误，Stream closed by remote ● YY013：通信库错误，Wait poll unknown error ● YY014：快照非法，Snapshot invalid ● YY015：连接获取错误，Connection receive wrong ● 53200：内存耗尽，Out of memory ● 08006：GTM出错，Connection failure ● 08000：连接出现错误，和DN的通讯失败，Connection exception ● 57P01：管理员关闭系统，Admin shutdown	retry_times取值范围为：5-10

名称	命令说明	取值范围
	● XX003：关闭远程套接字，Stream remote close socket ● XX009：重复查询，Duplicate query id ● YY016：stream查询并发更新同一行，Stream concurrent update ● CG003：内存分配错误， Allocate error ● CG004：致命错误，Fatal error ● F0011：临时文件读取错误，File error 同时，出错时gsql会查询所有CN/DN的连接状态，当状态异常时会sleep 1分钟再进行重试，能够覆盖大部分主备切换场景下的出错重试。 **说明** 1. 不支持事务块中的语句错误重试； 2. 不支持通过ODBC、JDBC接口查询的出错重试； 3. 含有unlogged表的sql语句，不支持节点故障后的出错重试； 4. 当前不支持CN和GTM节点故障时，gsql客户端的出错重试； 5. gsql客户端本身出现的错误，不在重跑考虑范围之内；	

表 3-27 大对象元命令

参数	参数说明
\lo_list	显示一个目前存储在该数据库里的所有DWS大对象及其说明。

表 3-28 流程控制元命令

参数	参数说明	取值范围
\if EXPR \elif EXPR \else \endif	这组元命令可实现可嵌套的条件块： • 条件块以\if开始，\endif结束。 • \if和\endif两者之间可以出现任意数量的\elif子句，或单一的\else子句。 • \if和\elif命令支持布尔表达式计算和字符串等值判断。 • \elif不能出现在\else和\endif之间。	• 布尔表达式计算和gsql原有方式保持一致：true/false、yes/no、on/off、1/0，其他被认为是true。 • 操作符和字符串之间必须使用空格。 • 支持数字比较和字符串比较，比较规则由固有变量COMPARE_STRATEGY控制（详见**表3-2**）。默认比较规则中，使用单引号界定字符串和数字。不同规则使用示例详见**\if条件块比较规则说明与示例**。 • 支持大小比较和等值判断，支持的操作符有：<，<=，>，>=，==，!=和<>。
\goto LABEL \label LABEL	这组元命令可实现无条件跳转： • \label元命令用于创建标签。 • \goto元命令用于跳转，可实现向上跳转和向下跳转。 **说明** • 交互模式不支持使用。 • \label元命令不支持在\if语句块中使用。 • \goto \label不建议在事务、PL/SQL等语句块中使用，避免出现不可预期结果。	• LABEL大小写敏感。 • LABEL只能包含大小写字母、数字和下划线。 • LABEL最大长度限制为32（含'\0'），如果超过长度限制，会截断使用，并输出告警信息。 • \label后面的标签名如果在同一个session中重复出现，会报错。

参数	参数说明	取值范围
\for \loop \exit-for \end-for	这组元命令可实现循环： • 循环块以\for开始，以\end-for结束。 • \for和\loop之间为循环条件，只支持SQL语句，不支持变量迭代，例如\for (i=0; i<100; ++i)。 • 循环条件中出现多条SQL语句时，以最后一条SQL语句的执行结果为循环条件。作为循环条件的SQL语句不能以分号结尾。 • \loop和\end-for之间为循环体，可以使用\exit-for退出循环。 • \for循环块支持多层嵌套。 **说明** • 交互模式不支持使用。 • 不支持在\parallel中使用\for循环块。 • \for循环块不建议在事务、PL/SQL等语句块中使用，避免出现不可预期结果。 • \label元命令不支持在\for循环块中使用。 • \for \loop之间不支持使用匿名块。	-

流程控制元命令使用示例：

- \if条件块使用示例

  示例文件test.sql：

  ```
  SELECT 'Jack' AS "Name";

  \if ${ERROR}
     \echo 'An error occurred in the SQL statement'
     \echo ${LAST_ERROR_MESSAGE}
  \elif '${Name}' == 'Jack'
     \echo 'I am Jack'
  \else
     \echo 'I am not Jack'
  \endif
  ```

  gsql -d -p 25308 --dynamic-param -f test.sql 执行结果：

  ```
   Name
  ------
   Jack
  (1 row)

  I am Jack
  ```

  上面的执行结果表示，第一个SQL语句执行成功，并设置Name变量，所以进入\elif分支，输出“I am Jack”。特殊变量ERROR和LAST_ERROR_MESSAGE的使用参见**表3-2**。

- \if条件块比较规则说明与示例
  - default：默认的比较策略，只支持字符串或数字比较，不支持混合比较。单引号内的按照字符串处理，单引号外的按照数字处理。

    示例文件test.sql：

    ```
    \set Name 'Jack'
    \set ID 1002

    -- 以单引号界定，在单引号内的使用字符串比较
    \if '${Name}' != 'Jack'
        \echo 'I am not Jack'
    -- 没有单引号，使用数字比较
    \elif ${ID} > 1000
        \echo 'Jack\'id is bigger than 1000'
    \else
        \echo 'error'
    \endif
    ```

    gsql -d -p 25308 --dynamic-param -f test.sql 执行结果：

    ```
    Jack'id is bigger than 1000
    ```

    如果使用操作符两侧，一侧使用了单引号，一侧未使用，认定为字符串和数字比较。不支持，则报错。

    ```
    postgres=> \set Name 'Jack'
    postgres=> \if ${Name} == 'Jack'
    ERROR: left[Jack] is a string without quote or number, and right['Jack'] is a string with quote, \if or \elif does not support this expression.
    WARNING: The input with quote is treated as a string, and the input without quote is treated as a number.
    postgres@> \endif
    ```

  - natural：在default的基础上，包含动态变量的也按照字符串处理。当比较操作符有一侧是数字比较，尝试将另一侧转换为数字，然后比较。如果转换失败，报错且比较结果为假。
    - 识别为字符串的条件有两个，满足任何一个即可。条件一，使用单引号，如'Jack'；条件二，字符串中包含动态变量（ ${VAR}和:VAR两种 ），不论变量是否存在，如${Name}_data。条件一和条件二同时满足，如'${Name}_data'。
    - 无法识别为字符串的，尝试数字识别。如无法转换成数字，则报错，如1011Q1没有使用单引号、不包含动态变量且无法转换成数字。
    - 如果比较符的两侧有一侧未识别为字符串或者数字，无法进行比较，则报错。
    - 如果比较符的一侧识别为数字，按照数字比较，如果另一侧无法转换为数字，则报错。
    - 如果比较符的两侧都识别为字符串，按照字符串比较。

    字符串比较，示例文件test.sql：

    ```
    \set COMPARE_STRATEGY natural
    SELECT 'Jack' AS "Name";

    -- 与'${Name}' > 'Jack'效果等同
    \if ${Name} == 'Jack'
        \echo 'I am Jack'
    \else
        \echo 'I am not Jack'
    \endif
    ```

    gsql -d -p 25308 --dynamic-param -f test.sql 执行结果：

```
 Name
------
 Jack
(1 row)

I am Jack
```

数字比较，示例文件test.sql：

```
\set COMPARE_STRATEGY natural
SELECT 1022 AS id;

-- 如果使用${id} == '01022'，则结果是不等，因为两侧都是字符串，使用字符串比较，结果为不等
\if ${id} == 01022
    \echo 'id is 1022'
\else
    \echo 'id is not 1022'
\endif
```

gsql -d -p 25308 --dynamic-param -f test.sql 执行结果：

```
 id
------
 1022
(1 row)

id is 1022
```

错误比较示例：

```
-- 操作符有一侧无法识别为字符串或数字
postgres=> \set COMPARE_STRATEGY natural
postgres=> \if ${Id} > 123sd
ERROR: The right[123sd] can not be treated as a string or a number. A numeric string should contain only digits and one decimal point, and a string should be enclosed in quote or contain dynamic variables, please check it.
-- 操作符一侧数字无法正确转换
postgres=> \set COMPARE_STRATEGY natural
postgres=> \if ${Id} <> 11101.1.1
ERROR: The right[11101.1.1] can not be treated as a string or a number. A numeric string should contain only digits and one decimal point, and a string should be enclosed in quote or contain dynamic variables, please check it.
```

- equal：只支持等值比较，所有情况按照字符串比较。

  示例文件test.sql：

```
\set COMPARE_STRATEGY equal
SELECT 'Jack' AS "Name";

\if ${ERROR}
    \echo 'An error occurred in the SQL statement'
-- equal比较规则下只支持字符串等值判断，大小比较直接报错，无定界符。下面的效果与${Name} == Jack等价
\elif '${Name}' == 'Jack'
    \echo 'I am Jack'
\else
    \echo 'I am not Jack'
\endif
```

  gsql -d -p 25308 --dynamic-param -f test.sql 执行结果：

```
 Name
------
 Jack
(1 row)

I am Jack
```

- \goto \label跳转示例

  示例文件test.sql：

```
\set Name Tom

\goto TEST_LABEL
```

```
SELECT 'Jack' AS "Name";

\label TEST_LABEL
\echo ${Name}
```

gsql -d -p 25308 --dynamic-param -f test.sql 执行结果：

```
Tom
```

上面的执行结果表示，\goto元命令实现跳转，直接执行\echo命令，没有对变量Name重新赋值。

- \if条件块和\goto \label结合使用示例

  示例文件test.sql：

```
\set Count 1

\label LOOP
\if ${Count} != 3
    SELECT ${Count} + 1 AS "Count";
    \goto LOOP
\endif

\echo Count = ${Count}
```

  gsql -d -p 25308 --dynamic-param -f test.sql 执行结果：

```
 Count
-------
     2
(1 row)

 Count
-------
     3
(1 row)

Count = 3
```

  上面的执行结果表示，通过\if条件块和\goto \label的结合实现简单的循环。

- \for循环块使用示例

  为了展示该功能，示例数据如下：

```
create table student (id int, name varchar(32));
insert into student values (1, 'Jack');
insert into student values (2, 'Tom');
insert into student values (3, 'Jerry');
insert into student values (4, 'Danny');

create table course (class_id int, class_day varchar(5), student_id int);
insert into course values (1004, 'Fri', 2);
insert into course values (1003, 'Tue', 1);
insert into course values (1003, 'Tue', 4);
insert into course values (1002, 'Wed', 3);
insert into course values (1001, 'Mon', 2);
```

  \for循环使用示例文件test.sql：

```
\for
select id, name from student order by id limit 3 offset 0
\loop
    \echo -[ RECORD ]+-----
    \echo id '\t'| ${id}
    \echo name '\t'| ${name}
\end-for
```

  gsql -d -p 25308 --dynamic-param -f test.sql 执行结果：

```
-[ RECORD ]+-----
id      | 1
name    | Jack
-[ RECORD ]+-----
id      | 2
```

```
name    | Tom
-[ RECORD ]+-----
id      | 3
name    | Jerry
```

上面的执行结果表示，通过循环块对SQL语句的执行结果进行遍历，\loop和\end-for之间可以出现更多语句，实现复杂的逻辑。

如果作为循环条件的SQL语句执行失败或者结果集为空，\loop和\end-for之间的语句将不被执行。

示例文件test.sql：

```
\for
select id, name from student_error order by id limit 3 offset 0
\loop
    \echo -[ RECORD ]+-----
    \echo id '\t'| ${id}
    \echo name '\t'| ${name}
\end-for
```

gsql -d -p 25308 --dynamic-param -f test.sql 执行结果：

```
gsql:test.sql:3: ERROR:  relation "student_error" does not exist
LINE 1: select id, name from student_error order by id limit 3 offset...
                             ^
```

上面的执行结果表示，student_error这个表不存在，所以SQL语句执行失败，\loop和\end-for之间的语句将不被执行。

- \exit-for退出循环

  示例文件test.sql：

  ```
  \for
  select id, name from student order by id
  \loop
      \echo ${id} ${name}
      \if ${id} == 2
          \echo find id(2), name is ${name}
          \exit-for
      \endif
  \end-for
  ```

  gsql -d -p 25308 --dynamic-param -f test.sql 执行结果：

  ```
  1 Jack
  2 Tom
  find id(2), name is Tom
  ```

  表student中的数据超过两行，当id=2时，使用\exit-for退出循环，不再继续执行。这个过程中也有与\if条件块的配合使用。

- \for循环嵌套

  示例文件test.sql：

  ```
  \for
  select id, name from student order by id limit 2 offset 0
  \loop
      \echo ${id} ${name}
      \for
      select
      class_id, class_day
      from course
      where student_id = ${id}
      order by class_id
      \loop
          \echo '  '${class_id}, ${class_day}
      \end-for
  \end-for
  ```

  gsql -d -p 25308 --dynamic-param -f test.sql 执行结果：

  ```
  1 Jack
    1003, Tue
  ```

```
2 Tom
  1001, Mon
  1004, Fri
```

通过两层循环获得Jack、Tom相关的course表中的信息。

PATTERN

很多\d命令都可以用一个PATTERN参数来指定要被显示的对象名称。在最简单的情况下，PATTERN正好就是该对象的准确名称。在PATTERN中的字符通常会被变成小写形式（就像在SQL名称中那样），例如\dt FOO将会显示名为foo的表。就像在SQL名称中那样，把PATTERN放在双引号中可以阻止它被转换成小写形式。如果需要在一个PATTERN中包括一个真正的双引号字符，则需要把它写成两个相邻的双引号，这同样是符合SQL引用标识符的规则。例如，\dt "FOO""BAR"将显示名为FOO"BAR（不是foo"bar）的表。和普通的SQL名称规则不同，不能只在PATTERN的一部分周围放上双引号，例如\dt FOO"FOO"BAR将会显示名为fooFOObar的表。

不使用PATTERN参数时，\d命令会显示当前schema搜索路径中可见的全部对象——这等价于用*作为PATTERN。所谓对象可见是指可以直接用名称引用该对象，而不需要用schema来进行限定。要查看数据库中所有的对象而不管它们的可见性，可以把*.*用作PATTERN。

如果放在一个PATTERN中，*将匹配任意字符序列（包括空序列），而?会匹配任意的单个字符（这种记号方法就像 Unix shell 的文件名PATTERN一样）。例如，\dt int*会显示名称以int开始的表。但是如果被放在双引号内，*和?就会失去这些特殊含义而变成普通的字符。

包含一个点号（.）的PATTERN被解释为一个schema名称模式后面跟上一个对象名称模式。例如，\dt foo*.*bar*会显示名称以foo开始的schema中所有名称包括bar的表。如果没有出现点号，那么模式将只匹配当前schema搜索路径中可见的对象。同样，双引号内的点号会失去其特殊含义并且变成普通的字符。

高级用户可以使用字符类等正则表达式记法，如[0-9]可以匹配任意数字。所有的正则表达式特殊字符都按照《开发指南》中的POSIX正则表达式所说的工作。以下字符除外：

- .会按照上面所说的作为一种分隔符。
- *会被翻译成正则表达式记号.*。
- ?会被翻译成.。
- $则按字面意思匹配。

根据需要，可以通过书写?、(*R*+|)、(*R*|)和*R*?来分别模拟PATTERN字符.、*R**和*R*?。$不需要作为一个正则表达式字符，因为PATTERN必须匹配整个名称，而不是像正则表达式的常规用法那样解释（换句话说，$会被自动地追加到PATTERN上）。如果不希望该PATTERN的匹配位置被固定，可以在开头或者结尾写上*。注意在双引号内，所有的正则表达式特殊字符会失去其特殊含义并且按照其字面意思进行匹配。另外，在操作符名称PATTERN中（即\do的PATTERN参数），正则表达式特殊字符也按照字面意思进行匹配。

3.7 常见问题处理

连接性能问题

- 数据库内核执行初始化语句较慢导致的性能问题。

此种情况定位较难，可以尝试使用Linux的跟踪命令：strace。

```
strace gsql -U MyUserName -W {password} -d postgres -h 127.0.0.1 -p 23508 -r -c '\q'
```

此时便会在屏幕上打印出数据库的连接过程。比如较长时间停留在下面的操作上：

```
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}])
```

此时便可以确定是数据库执行"SELECT VERSION()"语句较慢。

在连接上数据库后，便可以通过执行“explain performance select version()”语句来确定初始化语句执行较慢的原因。更多信息请参考《开发指南》中的“SQL执行计划介绍”章节。

另外还有一种场景不太常见：由于数据库CN所在机器的磁盘满或故障，此时所查询等受影响，无法进行用户认证，导致连接过程挂起，表现为假死。解决此问题清理数据库CN的数据盘空间便可。

- TCP连接创建较慢问题。

  此问题可以参考上面的初始化语句较慢排查的做法，通过strace跟踪，如果长时间停留在：

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

  或者

  ```
  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)
  ```

  那么说明客户端与数据库端建立物理连接过慢，此时应当检查网络是否存在不稳定、网络吞吐量太大的问题。

创建连接故障

- gsql: could not connect to server: No route to host

  此问题一般是指定了不可达的地址或者端口导致的。请检查-h参数与-p参数是否添加正确。

- gsql: FATAL: Invalid username/password,login denied.

  此问题一般是输入了错误的用户名和密码导致的，请联系数据库管理员，确认用户名和密码的正确性。

- The "libpq.so" loaded mismatch the version of gsql, please check it.

  此问题是由于环境中使用的libpq.so的版本与gsql的版本不匹配导致的，请通过“ldd gsql”命令确认当前加载的libpq.so的版本，并通过修改LD_LIBRARY_PATH环境变量来加载正确的libpq.so。

- gsql: symbol lookup error: xxx/gsql: undefined symbol: libpqVersionString

  此问题是由于环境中使用的libpq.so的版本与gsql的版本不匹配导致的（也有可能是环境中存在PostgreSQL的libpq.so），请通过“ldd gsql”命令确认当前加载的libpq.so的版本，并通过修改LD_LIBRARY_PATH环境变量来加载正确的libpq.so。

- 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?

  此问题是由于网络连接故障造成。请检查客户端与数据库服务器间的网络连接。如果发现从客户端无法PING到数据库服务器端，则说明网络连接出现故障。请联系网络管理人员排查解决。

  ```
  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.

  此问题是由于用户不具备访问该数据库的权限，可以使用如下方法解决。

  a. 使用管理员用户dbadmin连接数据库。

  ```
  gsql -d postgres -U dbadmin -p 8000
  ```

  b. 赋予该用户访问数据库的权限。

  GRANT CONNECT ON DATABASE postgres TO user1;

  说明

  实际上，常见的许多错误操作也可能产生用户无法连接上数据库的现象。如用户连接的数据库不存在，用户名或密码输入错误等。这些错误操作在客户端工具也有相应的提示信息。

  ```
  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.

  此问题是由于系统连接数量超过了最大连接数量。请联系数据库DBA进行会话连接数管理，释放无用会话。

  关于查看用户会话连接数的方法如表3-29。

  会话状态可以在视图PG_STAT_ACTIVITY中查看。无用会话可以使用函数pg_terminate_backend进行释放。

  ```
  select datid,pid,state from pg_stat_activity;
   datid |       pid       | state
  -------+-----------------+--------
   13205 | 139834762094352 | active
   13205 | 139834759993104 | idle
  (2 rows)
  ```

  其中pid的值即为该会话的线程ID。根据线程ID结束会话。

  ```
  SELECT PG_TERMINATE_BACKEND(139834759993104);
  ```

  显示类似如下信息，表示结束会话成功。

  ```
  PG_TERMINATE_BACKEND
  ----------------------
   t
  (1 row)
  ```

表 3-29 查看会话连接数

描述	命令
查看指定用户的会话连接数上限。	执行如下命令查看连接到指定用户USER1的会话连接数上限。其中-1表示没有对用户user1设置连接数的限制。 SELECT ROLNAME,ROLCONNLIMIT FROM PG_ROLES WHERE ROLNAME='user1'; rolname | rolconnlimit ---------+-------------- user1 | -1 (1 row)
查看指定用户已使用的会话连接数。	执行如下命令查看指定用户USER1已使用的会话连接数。其中，1表示USER1已使用的会话连接数。 SELECT COUNT(*) FROM V$SESSION WHERE USERNAME='user1'; count ------- 1 (1 row)
查看指定数据库的会话连接数上限。	执行如下命令查看连接到指定数据库postgres的会话连接数上限。其中-1表示没有对数据库postgres设置连接数的限制。 SELECT DATNAME,DATCONNLIMIT FROM PG_DATABASE WHERE DATNAME='postgres'; datname | datconnlimit ----------+-------------- postgres | -1 (1 row)
查看指定数据库已使用的会话连接数。	执行如下命令查看指定数据库postgres上已使用的会话连接数。其中，1表示数据库postgres上已使用的会话连接数。 SELECT COUNT(*) FROM PG_STAT_ACTIVITY WHERE DATNAME='postgres'; count ------- 1 (1 row)
查看所有用户已使用会话连接数。	执行如下命令查看所有用户已使用的会话连接数。 SELECT COUNT(*) FROM V$SESSION; count ------- 10 (1 row)

- gsql: wait xxx.xxx.xxx.xxx:xxxx timeout expired

  gsql在向数据库发起连接的时候，会有5分钟超时机制，如果在这个超时时间内，数据库未能正常的对客户端请求进行校验和身份认证，那么gsql会退出当前会话的连接过程，并报出如上错误。

  一般来说，此问题是由于连接时使用的-h参数及-p参数指定的连接主机及端口有误（即错误信息中的xxx部分），导致通信故障；极少数情况是网络故障导致。要排除此问题，请检查数据库的主机名及端口是否正确。

- gsql: could not receive data from server: Connection reset by peer.

同时，检查CN日志中出现类似如下日志“ FATAL: cipher file "/data/coordinator/server.key.cipher" has group or world access”，一般是由于数据目录或部分关键文件的权限被误操作篡改导致。请参照其他正常实例下的相关文件权限，修改回来便可。

- gsql: FATAL: GSS authentication method is not allowed because XXXX user password is not disabled.

  目标CN的pg_hba.conf里配置了当前客户端IP使用"gss"方式来做认证，该认证算法不支持用作客户端的身份认证，请修改到"sha256"后再试。

  说明

  - 请不要修改pg_hba.conf中数据库集群主机的相关设置，否则可能导致数据库功能故障。
  - 建议业务应用部署在数据库集群之外，而非集群内部。

其他故障

- 出现因“总线错误”（Bus error）导致的core dump或异常退出

  一般情况下出现此种问题，是进程运行过程中加载的共享动态库（在Linux为.so文件）出现变化；或者进程二进制文件本身出现变化，导致操作系统加载机器的执行码或者加载依赖库的入口发生变化，操作系统出于保护目的将进程终止，产生core dump文件。

  解决此问题，重试便可。同时请尽可能避免在升级等运维操作过程中，在集群内部运行业务程序，避免升级时因替换文件产生此问题。

  说明

  此故障的core dump文件的可能堆栈是dl_main及其子调用，它是操作系统用来初始化进程做共享动态库加载的。如果进程已经初始化，但是共享动态库还未加载完成，严格意义上来说，进程并未完全启动。

4 GDS

4.1 安装配置和启动 GDS

操作场景

DWS提供了数据服务工具GDS来帮助分发待导入的用户数据及实现数据的高速导入。GDS需部署到数据服务器上。

数据量大，数据存储在多个服务器上时，在每个数据服务器上安装配置、启动GDS后，各服务器上的数据可以并行入库。GDS在各台数据服务器上的安装配置和启动方法相同，本节以一台服务器为例进行说明。

背景信息

GDS的版本需与集群版本保持一致（如：GDS V100R008C00版本与DWS 1.3.X版本配套），否则可能会出现导入导出失败或导入导出进程停止响应等情况。因此请勿使用历史版本的GDS进行导入。

数据库版本升级后，请按照**操作步骤**中的办法下载DWS软件包解压缩自带的GDS进行安装配置和启动。在导入导出开始时，DWS也会进行两端的版本一致性检测，不一致时会在屏幕上显示报错信息并终止对应操作。

GDS的版本号的查看办法为：在GDS工具的解压目录下执行如下命令。

```
gds -V
```

数据库版本的查看办法为：连接数据库后，执行如下SQL命令查看。

```
SELECT version();
```

操作步骤

步骤1 以root用户登录待安装GDS的数据服务器，创建存放GDS工具包的目录。

```
mkdir -p /opt/bin/dws
```

步骤2 将GDS工具包上传至上一步所创建的目录中。

以上传SUSE Linux版本的工具包为例，将GDS工具包“dws_client_8.x.x_suse_x64.zip”上传至上一步所创建的目录中。

步骤3 （可选）如果使用SSL加密传输，请一并上传SSL证书至**步骤1**所创建的目录下。

步骤4 在工具包所在目录下，解压工具包。

```
cd /opt/bin/dws
unzip dws_client_8.x.x_suse_x64.zip
```

步骤5 创建GDS专有用户及其所属的用户组。此用户用于启动GDS及读取源数据。

```
groupadd gdsgrp
useradd -g gdsgrp gds_user
```

步骤6 分别修改工具包和数据源文件目录属主为GDS专有用户。

```
chown -R gds_user:gdsgrp /opt/bin/dws/gds
chown -R gds_user:gdsgrp /input_data
```

步骤7 切换到gds_user用户。

```
su - gds_user
```

若当前集群版本为8.0.x及以前版本，请跳过**步骤8**，直接执行**步骤9**。

若当前集群版本为8.1.x版本，则正常执行以下步骤。

步骤8 执行环境依赖脚本（仅8.1.x版本适用）。

```
cd /opt/bin/dws/gds/bin
source gds_env
```

步骤9 启动GDS服务。

GDS是绿色软件，解压后启动即可。GDS启动方式有两种：

方式一：直接使用“gds”命令，在命令项中设置启动参数。

方式二：将启动参数写进配置文件“gds.conf”后， 使用“gds_ctl.py”命令启动。

对于集中一次性导入的场景推荐使用第一种方式。对于需要隔段时间再次导入的场景，推荐使用第二种方式以配置文件的形式提升启动效率。

- 方式一：直接使用“gds”命令，启动GDS。
  - 非SSL模式传输数据的情况下，启动GDS。

    ```
    gds -d dir -p ip:port -H address_string -l log_file -D -t worker_num
    ```

    示例：

    ```
    /opt/bin/dws/gds/bin/gds -d /input_data/ -p 192.168.0.90:5000 -H 10.10.0.1/24 -
    l /opt/bin/dws/gds/gds_log.txt -D -t 2
    ```

  - 使用SSL加密方式传输数据的情况下，启动GDS。

    ```
    gds -d dir -p ip:port -H address_string -l log_file -D
    -t worker_num --enable-ssl --ssl-dir Cert_file
    ```

    示例：

    以**步骤3**中SSL证书已上传至/opt/bin为例，命令如下。

    ```
    /opt/bin/dws/gds/bin/gds -d /input_data/ -p 192.168.0.90:5000 -H 10.10.0.1/24 -
    l /opt/bin/dws/gds/gds_log.txt -D --enable-ssl --ssl-dir /opt/bin/
    ```

  命令中的斜体部分请根据实际替换。

  - **-d** *dir*：保存有待导入数据的数据文件所在目录。本教程中为“/input_data/”。
  - **-p** *ip:port*：GDS监听IP和监听端口。默认值为：127.0.0.1，需要替换为能跟DWS通信的万兆网IP。监听端口的取值范围：1024~65535。默认值为：8098。本教程配置为：192.168.0.90:5000。
  - **-H** *address_string*：允许哪些主机连接和使用GDS服务。参数需为CIDR格式。此参数配置的目的是允许DWS集群可以访问GDS服务进行数据导入。所以请保证所配置的网段包含DWS集群各主机。

- **-l** *log_file*：存放GDS的日志文件路径及文件名。本教程为“/opt/bin/dws/gds/gds_log.txt”。
- **-D**：后台运行GDS。仅支持Linux操作系统下使用。
- **-t** *worker_num*：设置GDS并发线程数。默认值为：8。取值范围为0<worker_num<=200，正整数。DWS及数据服务器上的I/O资源均充足时，可以加大并发线程数。

  GDS是根据导入事务并发数来决定服务运行线程数的。也就是说即使启动GDS时设置了多线程，也并不会加速单个导入事务。未做过人为事务处理时，一条INSERT语句就是一个导入事务。
- **--enable-ssl**：启用SSL加密方式传输数据。
- **--ssl-dir** *Cert_file*：SSL证书所在目录。需与**步骤3**中的证书保存目录保持一致。

关于更多参数的设置信息请参考《数据仓库服务工具指南》中的“GDS并行数据加载工具 > gds命令简介”。

- 方式二：将启动参数写进配置文件“gds.conf”后， 使用“gds_ctl.py”命令启动。

  a. 使用如下命令，进入GDS工具包的“config”目录下，配置“gds.conf”文件。“gds.conf”配置详细信息请参考**表4-1**。

  ```
  vim /opt/bin/dws/gds/config/gds.conf
  ```

  示例：

  配置“gds.conf”文件如下：

  ```
  <?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>
  ```

  配置文件信息如下：

  - 数据服务器所在IP为192.168.0.90，GDS监听端口为5000。
  - 数据文件存放在“/input_data/”目录下。
  - 错误日志文件存放在“/err”目录下。该目录需要拥有GDS读写权限的用户自行创建。
  - 单个数据文件大小为100MB。
  - 每个错误日志大小为100MB。
  - 日志保存在“/log/gds_log.txt”文件中。该目录需要拥有GDS读写权限的用户自行创建。
  - 只允许IP为10.10.0.*的节点进行连接。
  - GDS进程以后台方式运行。
  - 递归数据文件目录。
  - 指定并发导入工作线程数目为2。

  b. 执行如下命令启动GDS并确认GDS是否启动成功。

  ```
  python3 gds_ctl.py start
  ```

示例：

```
cd /opt/bin/dws/gds/bin
python3 gds_ctl.py start
Start GDS gds1                  [OK]
gds [options]:
 -d dir             Set data directory.
 -p port             Set GDS listening port.
    ip:port         Set GDS listening ip address and port.
 -l log_file        Set log file.
 -H secure_ip_range
                   Set secure IP checklist in CIDR notation. Required for GDS to start.
 -e dir             Set error log directory.
 -E size            Set size of per error log segment.(0 < size < 1TB)
 -S size            Set size of data segment.(1MB < size < 100TB)
 -t worker_num      Set number of worker thread in multi-thread mode, the upper limit is 200. If
without setting, the default value is 8.
 -s status_file     Enable GDS status report.
 -D                 Run the GDS as a daemon process.
 -r                 Read the working directory recursively.
 -h                 Display usage.
```

----结束

gds.conf 参数说明

表 4-1 gds.conf 配置说明

属性	说明	取值范围
name	标识名。	-
ip	监听ip地址。	IP需为合法IP地址。 IP的默认值：127.0.0.1
port	监听端口号。	取值范围：1024~65535，正整数。 默认值：8098。
data_dir	数据文件目录。	-
err_dir	错误日志文件目录。	默认值：数据文件目录
log_file	日志文件路径。	-
host	设置允许连接到GDS的主机IP地址（参数为CIDR格式，仅支持linux系统）。	-
recursive	是否递归数据文件目录。选择true时会递归读取location指定的目录层级下所有的同名文件。	取值范围： ● true：递归 。 ● false：不递归。 默认值：false。
daemon	是否以DAEMON（后台）模式运行。	取值范围： ● true：以DAEMON模式运行。 ● false：不以DAEMON模式运行。 默认值：false。

属性	说明	取值范围
parallel	导入工作线程并发数目。	取值范围：0~200，正整数。 默认值：8。

4.2 停止 GDS

操作场景

待导入数据成功后，停止GDS。

操作步骤

步骤1 以gds_user用户登录安装GDS的数据服务器。

步骤2 请根据启动GDS的方式，选择停止GDS的方式。

- 若用户使用“gds”命令启动GDS，请使用以下方式停止GDS。
  a. 执行如下命令，查询GDS进程号。
  ```
  ps -ef|grep gds
  ```
  示例：其中GDS进程号为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 -l /log/gds_log.txt -D
  gds_user 129003 118723  0 15:04 pts/0    00:00:00 grep gds
  ```
  b. 使用“kill”命令，停止GDS。其中128954为上一步骤中查询出的GDS进程号。
  ```
  kill -9 128954
  ```

----结束

4.3 GDS 导入示例

示例：多数据服务器并行导入

规划数据服务器与集群处于同一内网，数据服务器IP为192.168.0.90和192.168.0.91。数据源文件格式为CSV。

1. 创建导入的目标表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. 以root用户登录每台GDS数据服务器，在两台数据服务器上，分别创建数据文件存放目录“/input_data”。以下以IP为192.168.0.90的数据服务器为例进行操作，剩余服务器上的操作与它一致。
   ```
   mkdir -p /input_data
   ```
3. （可选）创建用户及其所属的用户组。此用户用于启动GDS。若该类用户及所属用户组已存在，可跳过此步骤。

```
groupadd gdsgrp
useradd -g gdsgrp gds_user
```

4. 将数据源文件均匀分发至相应数据服务器的“/input_data”目录中。
5. 修改每台数据服务器上数据文件及数据文件目录“/input_data”的属主为gds_user。以下以IP为192.168.0.90的数据服务器为例，进行操作。

```
chown -R gds_user:gdsgrp /input_data
```

6. 以gds_user用户登录每台数据服务器上分别启动GDS。

   其中GDS安装路径为“/opt/bin/dws/gds”，数据文件存放在“/input_data/”目录下，数据服务器所在IP为192.168.0.90和192.168.0.91，GDS监听端口为5000，以后台方式运行。

   在IP为192.168.0.90的数据服务器上启动GDS。

```
/opt/bin/dws/gds/gds -d /input_data -p 192.168.0.90:5000 -H 10.10.0.1/24 -D
```

   在IP为192.168.0.91的数据服务器上启动GDS。

```
/opt/bin/dws/gds/gds -d /input_data -p 192.168.0.91:5000 -H 10.10.0.1/24  -D
```

7. 创建外表tpcds.foreign_tpcds_reasons用于接收数据服务器上的数据。

   其中设置**导入模式信息**如下所示：

   - 导入模式为Normal模式。
   - 由于启动GDS时，设置的数据源文件存放目录为“/input_data”，GDS监听端口为5000，所以设置参数“location”为“gsfs://192.168.0.90:5000/* | gsfs://192.168.0.91:5000/*”。

   设置**数据格式信息**是根据导出时设置的详细数据格式参数信息指定的，参数设置如下所示：

   - 数据源文件格式（format）为CSV。
   - 编码格式（encoding）为UTF-8。
   - 字段分隔符（delimiter）为E'\x08'。
   - 引号字符（quote）为0x1b。
   - 数据文件中空值（null）为没有引号的空字符串。
   - 逃逸字符（escape）默认和quote相同。
   - 数据文件是否包含标题行（header）为默认值false，即导入时数据文件第一行被识别为数据。

   设置**导入容错性**如下所示：

   - 允许出现的数据格式错误个数（PER NODE REJECT LIMIT 'value'）为unlimited，即接受导入过程中所有数据格式错误。
   - 将数据导入过程中出现的数据格式错误信息（LOG INTO error_table_name）写入表err_tpcds_reasons。

   根据以上信息，创建的外表如下所示：

```
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. 通过外表tpcds.foreign_tpcds_reasons，将数据导入目标表tpcds.reasons。

```
INSERT INTO tpcds.reasons SELECT * FROM tpcds.foreign_tpcds_reasons;
```

9. 查询错误信息表err_tpcds_reasons，处理数据导入错误。详细请参见**处理错误表**。

```
SELECT * FROM err_tpcds_reasons;
```

10. 待数据导入完成后，以gds_user用户登录每台数据服务器，分别停止GDS。

    以下以IP为192.168.0.90的数据服务器为例，停止GDS。其中GDS进程号为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
```

示例：多线程导入

规划数据服务器与集群处于同一内网，数据服务器IP为192.168.0.90，导入的数据源文件格式为CSV，同时导入2个目标表。

1. 在数据库中创建导入的目标表tpcds.reasons1和tpcds.reasons2。

```
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. 以root用户登录GDS数据服务器，创建数据文件存放目录“/input_data”，以及子目录“/input_data/import1/”和“/input_data/import2/”。

```
mkdir -p /input_data
```

3. 将目标表tpcds.reasons1的数据源文件存放在数据服务器“/input_data/import1/”目录下，将目标表tpcds.reasons2的数据源文件存放在目录“/input_data/import2/”下。

4. （可选）创建用户及其所属的用户组。此用户用于启动GDS。若该用户及所属用户组已存在，可跳过此步骤。

```
groupadd gdsgrp
useradd -g gdsgrp gds_user
```

5. 修改数据服务器上数据文件及数据文件目录“/input_data”的属主为gds_user。

```
chown -R gds_user:gdsgrp /input_data
```

6. 以gds_user用户登录数据服务器上启动GDS。

    其中GDS安装路径为“/gds”，数据文件存放在“/input_data/”目录下，数据服务器所在IP为192.168.0.90，GDS监听端口为5000，以后台方式运行，设定并发度为2，并设定递归文件目录。

```
/gds/gds -d /input_data -p 192.168.0.90:5000 -H 10.10.0.1/24  -D -t 2 -r
```

7. 在数据库中创建外表tpcds.foreign_tpcds_reasons1和tpcds.foreign_tpcds_reasons2用于接收数据服务器上的数据。

    以下以外表tpcds.foreign_tpcds_reasons1为例，讲解设置的导入外表参数信息。

    其中设置的**导入模式信息**如下所示：

    - 导入模式为Normal模式。
    - 由于启动GDS时，设置的数据源文件存放目录为“/input_data/”，GDS监听端口为5000，实际存放数据源文件目录为“/input_data/import1/”，所以设置参数“location”为“gsfs://192.168.0.90:5000/import1/*”。

    设置的**数据格式信息**是根据导出时设置的详细数据格式参数信息指定的，参数设置如下所示：

- 数据源文件格式（format）为CSV。
- 编码格式（encoding）为UTF-8。
- 字段分隔符（delimiter）为E'\x08'。
- 引号字符（quote）为0x1b。
- 数据文件中空值（null）为没有引号的空字符串。
- 逃逸字符（escape）默认和quote相同。
- 数据文件是否包含标题行（header）为默认值false，即导入时数据文件第一行被识别为数据。

设置的**导入容错性**如下所示：

- 允许出现的数据格式错误个数（PER NODE REJECT LIMIT 'value'）为unlimited，即接受导入过程中所有数据格式错误。
- 将数据导入过程中出现的数据格式错误信息（LOG INTO error_table_name）写入表err_tpcds_reasons1。
- 当数据源文件中一行的最后一个字段缺失（fill_missing_fields）时，自动设置为NULL。

根据以上信息，创建的外表tpcds.foreign_tpcds_reasons1如下所示：

```
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';
```

参考以上设置，创建的外表tpcds.foreign_tpcds_reasons2如下所示：

```
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. 通过外表tpcds.foreign_tpcds_reasons1和tpcds.foreign_tpcds_reasons2将数据分别导入tpcds.reasons1和tpcds.reasons2。

```
INSERT INTO tpcds.reasons1 SELECT * FROM tpcds.foreign_tpcds_reasons1;
INSERT INTO tpcds.reasons2 SELECT * FROM tpcds.foreign_tpcds_reasons2;
```

9. 查询错误信息表err_tpcds_reasons1和err_tpcds_reasons2，处理数据导入错误。详细请参见**处理错误表**。

```
SELECT * FROM err_tpcds_reasons1;
SELECT * FROM err_tpcds_reasons2;
```

10. 待数据导入完成后，以gds_user用户登录数据服务器，停止GDS。

    其中GDS进程号为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
```

4.4 gds

背景信息

gds可以为DWS提供导入导出数据的功能。更多详细内容可参考《开发指南》的“导入数据”和“导出数据”章节。

语法

```
gds [ OPTION ] -d DIRECTORY
```

其中，-d、-H是必选参数，option项是可选参数。gds将DIRECTORY中的文件数据提供给DWS访问。

在启动GDS服务前， 请确定使用的GDS版本和数据库的版本保持一致，否则数据库会提示错误并终止导入导出操作，因此请注意GDS工具和数据库的版本务必严格匹配。具体版本可通过-V参数进行查看。

参数说明

- -d dir

  设置待导入数据文件的目录。在gds进程权限允许的条件下，-d指定的目录会自动被创建。

- -p ip:port

  设置gds监听IP和监听端口。

  IP的取值范围：IP需为合法IP地址。

  IP的默认值：127.0.0.1。

  监听端口的取值范围：1024~65535，正整数。

  监听端口的默认值：8098。

- -l log_file

  设置日志文件。本次特性添加了日志自动切分的功能。当设置-R参数后gds会根据设置的值重新生成新的文件，以此来避免单个日志文件过大的问题。

  生成规则：gds默认只识别后缀是log的文件重新生成日志文件。

  例如，当-l参数指定为 gds.log，-R指定为 20MB的时候，当gds.log大小达到20MB后就会新创建一个 "gds-2020-01-17_115425.log"文件。

  当-l指定的日志文件没有以log为后缀，例如：" gds.log.txt"， 则新创建的日志文件名为" gds.log-2020-01-19_122739.txt"。

  gds启动时会检测-l参数设置的日志文件是否存在，如果存在则根据当前日期时间新生成一个日志文件，不会覆盖之前的日志文件。

  说明

  如果当前目录下GDS进程的日志数目超过--log-filecount时，会触发旧日志的回收。为了保存大量日志，建议每个GDS设置单独目录，并且调高启动参数--log-filecount。

- -H address_string

  设置允许哪些主机连接到gds，参数需为CIDR格式，仅支持linux系统。需要配置多个不同网段时，使用“,”分隔。例如：-H 10.10.0.0/24,10.10.5.0/24。

- -e dir

  设置导入时产生的错误日志存放路径。

  默认值：数据文件目录。

- -E size

  设置导入产生的错误日志的上限值。

  取值范围：0<size<1TB，请使用正整数+单位的形式进行取值设置，单位支持KB、MB和GB。

- -S size

  设置导出单个文件大小上限。

  取值范围：1MB<size<100TB，请使用正整数+单位的形式进行取值设置，单位支持KB、MB和GB。如果使用KB，取值需要大于1024KB。

- -R size

  设置-l指定的gds单个日志文件大小上限。

  取值范围：1MB<size<1TB，请使用正整数+单位的形式进行取值设置，单位支持KB、MB和GB。如果使用KB，取值需要大于1024KB。

  默认值：16MB

- -t worker_num

  设置导入导出工作并发线程数目。

  取值范围：0<worker_num<=200，正整数

  默认值：8

  推荐值：普通文件导入导出场景取值：CPU核数*2；管道文件导入导出场景取值：64。

  说明

  当管道文件导入导出场景并发较大时，该值应不低于业务并发数。

- -s status_file

  设置状态文件，仅支持linux系统。

- -D

  后台运行gds，仅支持linux系统。

- -r

  递归遍历目录（外表目录下的子目录）下文件，会递归读取location指定的目录层级下所有的同名文件，仅支持linux系统。

- -h

  显示帮助信息。

- --enable-ssl

  使用SSL认证的方式与集群通信。

- --ssl-dir Cert_file

  在使用SSL认证方式时，指定认证证书的所在路径。

- --debug-level

  设置GDS端的debug日志级别，以控制GDS debug相关的日志输出。

  取值范围：0、1、2

  - 0：仅打印导入导出相关的文件列表，日志量小，推荐在系统处于正常状态时使用设置。
  - 1：打印日志的完整信息，增加各节点的连接信息、session转换信息和一些数据统计。
  - 2：打印详细的交互日志以及所属状态，输出较大量的debug日志信息，以帮助故障定位分析。推荐仅在故障定位时开启。

  默认值：1

- --log-filecount

  设置保留的日志文件最大个数，当日志文件个数超过该参数值，按照文件创建时间，保留最近创建的日志文件。

  取值范围：5<=log-filecount<=1024，正整数

  默认值：50

- --pipe-timeout

  设置GDS操作管道文件的等待超时时间。

  说明

  - 该参数的设置是为了避免人为或程序自身问题造成管道文件的一端长时间不读取或者不写入，导致管道另一端的读取或写入操作hang住。
  - 该参数表示的超时时间不是指GDS一个导入导出任务的最长时间，而是GDS对管道文件的每一次read/open/write的最大超时时间，当超过--pipe-timeout参数设置时间会向前端报错。

  取值范围：大于1s。请使用正整数+单位的形式进行取值设置，单位支持s、m和h。如：1小时可以设置为3600s、60m或者1h。

  默认值：1h/60m/3600s

- --pipe-size

  设置GDS管道文件导入/导出时所使用的文件容量。

  取值范围：大于1K。

  默认值：操作系统允许的最大值，可以通过命令 cat /proc/sys/fs/pipe-max-size 查看。

  说明

  该参数只能在Linux内核版本不低于2.6.35的环境下使用。

示例

数据文件存放在“/data” 目录，IP为192.168.0.90，监听端口为5000。

```
gds -d /data/ -p 192.168.0.90:5000 -H 10.10.0.1/24
```

数据文件存放在“/data/” 目录下的任意子目录，IP为192.168.0.90，监听端口为5000。

```
gds -d /data/ -p 192.168.0.90:5000 -H 10.10.0.1/24 -r
```

数据文件存放在“/data/ ”目录，IP为192.168.0.90，监听端口为5000，以后台方式运行，将日志保存在“/log/gds_log.txt”文件中，指定并发导入工作线程数目为32。

```
gds -d /data/ -p 192.168.0.90:5000 -H 10.10.0.1/24 -l /log/gds_log.txt -D  -t 32
```

数据文件存放在“/data/ ”目录，IP为192.168.0.90，监听端口为5000，只允许IP为10.10.0.*的节点进行连接。

```
gds -d /data/ -p 192.168.0.90:5000 -H 10.10.0.1/24
```

数据文件存放在“/data/ ”目录，IP为192.168.0.90，监听端口为5000，只允许IP为10.10.0.*的节点进行连接，设定为使用SSL认证的方式与集群通信，证书文件存放在/certfiles/目录。

```
gds -d /data/ -p 192.168.0.90:5000 -H 10.10.0.1/24 --enable-ssl --ssl-dir /certfiles/
```

说明

- 1个GDS在同一时刻，只能为1个集群提供导入导出服务；
- 为满足安全要求，请通过-p显示指定监听ip和监听端口。
- 证书文件包括根证书文件cacert.pem，以及二级证书文件client.crt和密钥文件client.key。
- 在加载证书时，需要使用密码保护文件client.key.rand和client.key.cipher。

4.5 gds_ctl.py

背景信息

在配置了gds.conf的情况下，就可通过gds_ctl.py控制gds的启动和停止。

前置条件

只支持在Linux系统执行该命令。执行前，需确保目录结构如下：

|----gds

|----gds_ctl.py

|----config

|-------gds.conf

|-------gds.conf.sample

或

|----gds

|----gds_ctl.py

|-------gds.conf

|-------gds.conf.sample

“gds.conf”的内容：

```
<?xml version="1.0"?>
<config>
<gds name="gds1" ip="127.0.0.1" port="8098" data_dir="/data" err_dir="/err" data_seg="100MB"
err_seg="1000MB" log_file="./gds.log" host="10.10.0.1/24" daemon='true' recursive="true" parallel="32"></
gds>
</config>
```

“gds.conf”配置说明：

- name：标识名。
- ip：监听ip地址。

- port：监听端口号。
  取值范围：1024~65535，正整数。
  默认值：8098。
- data_dir：数据文件目录。
- err_dir：错误日志文件目录。
- log_file：日志文件路径。
- host：允许哪些主机连接到gds。
- recursive：是否递归数据文件目录。
  取值范围：
  - true为递归数据文件目录。
  - false为不递归数据文件目录。
- daemon：是否以DAEMON模式运行，
  取值范围：
  - true为以DAEMON模式运行。
  - false为不以DAEMON模式运行。
- parallel：导入导出工作线程并发数目。
  默认并发数目为8，最大为200。

语法

```
gds_ctl.py [ start | stop all | stop [ ip: ] port | stop | status ]
```

描述

当配置了“gds.conf”，可通过gds_ctl.py启动/停止gds。

参数说明

- start
  启动gds.conf中配置的gds。
- stop
  关闭当前用户有权限关闭的经配置文件启动的gds运行实例。
- stop all
  关闭当前用户有权限关闭的所有gds运行实例。
- stop [ip:] port
  关闭当前用户有权限关闭的特定gds运行实例。如果启动时指定了ip:port，那么停止需要指定相应的ip:port；如果启动时未指定IP，只指定port，则停止只需指定相应的port即可。如启动和停止指定不同的信息，则停止失败。
- status
  查询通过gds.conf启动的gds实例的运行状态。

示例

启动gds。

```
python3 gds_ctl.py start
```

停止由配置文件启动的gds。

```
python3 gds_ctl.py stop
```

停止所有当前用户有权限关闭的gds。

```
python3 gds_ctl.py stop all
```

停止当前用户有权限关闭的，由[ip:]port指定的gds。

```
python3 gds_ctl.py stop 127.0.0.1:8098
```

查询gds状态。

```
python3 gds_ctl.py status
```

4.6 处理错误表

操作场景

当数据导入发生错误时，请根据本文指引信息进行处理。

查询错误信息

数据导入过程中发生的错误，一般分为数据格式错误和非数据格式错误。

- 数据格式错误

  在创建外表时，通过设置参数“LOG INTO error_table_name”，将数据导入过程中出现的数据格式错误信息写入指定的错误信息表error_table_name中。您可以通过以下SQL，查询详细错误信息。

  ```
  SELECT * FROM error_table_name;
  ```

  错误信息表结构如表4-2所示。

  表 4-2 错误信息表

列名称	类型	描述
nodeid	integer	报错节点编号。
begintime	timestamp with time zone	出现数据格式错误的时间。
filename	character varying	出现数据格式错误的数据源文件名。 当GDS导入时，同时会包括对应GDS服务端的IP地址端口信息。
rownum	bigint	在数据源文件中，出现数据格式错误的行号。
rawrecord	text	在数据源文件中，出现数据格式错误的原始记录。
detail	text	详细错误信息。

- 非数据格式错误

  对于非数据格式错误，一旦发生将导致整个数据导入失败。您可以根据执行数据导入过程中，界面提示的错误信息，帮助定位问题，处理错误表。

处理数据导入错误

根据获取的错误信息，请对照下表，处理数据导入错误。

表 4-3 处理数据导入错误

错误信息	原因	解决办法
missing data for column "r_reason_desc"	1. 数据源文件中的列数比外表定义的列数少。 2. 对于TEXT格式的数据源文件，由于转义字符（\）导致delimiter（分隔符）错位或者quote（引号字符）错位造成的错误。 **示例：**目标表存在3列字段，导入的数据如下所示。由于存在转义字符“\”，分隔符“|”被转义为第二个字段的字段值，导致第三个字段值缺失。 `BE|Belgium|1`	1. 由于列数少导致的报错，选择下列办法解决： ● 在数据源文件中，增加列“r_reason_desc”的字段值。 ● 在创建外表时，将参数“fill_missing_fields”设置为“on”。即在导入过程中，若数据源文件中一行数据的最后一个字段缺失，则把最后一个字段的值设置为NULL，不报错。 2. 对由于转义字符导致的错误，需检查报错的行中是否含有转义字符（\）。若存在，建议在创建外表时，将参数“noescaping”（是否不对'\'和后面的字符进行转义）设置为true。
extra data after last expected column	数据源文件中的列数比外表定义的列数多。	● 在数据源文件中，删除多余的字段值。 ● 在创建外表时，将参数“ignore_extra_data”设置为“on”。即在导入过程中，若数据源文件比外表定义的列数多，则忽略行尾多出来的列。
invalid input syntax for type numeric: "a"	数据类型错误。	在数据源文件中，修改输入字段的数据类型。根据此错误信息，请将输入的数据类型修改为numeric。
null value in column "staff_id" violates not-null constraint	非空约束。	在数据源文件中，增加非空字段信息。根据此错误信息，请增加“staff_id”列的值。

错误信息	原因	解决办法
duplicate key value violates unique constraint "reg_id_pk"	唯一约束。	• 删除数据源文件中重复的行。 • 通过设置关键字“DISTINCT”，从SELECT结果集中删除重复的行，保证导入的每一行都是唯一的。 **INSERT INTO** reasons **SELECT DISTINCT * FROM** foreign_tpcds_reasons;
value too long for type character varying(16)	字段值长度超过限制。	在数据源文件中，修改字段值长度。根据此错误信息，字段值长度限制为VARCHAR2(16)。

5 DSC

5.1 了解 DSC

当企业决定从传统数据库切换到华为云DWS数据库时，会遇到数据库迁移的挑战，特别是应用程序SQL脚本的迁移，这通常是一个复杂、高风险且耗时的过程。

如何确保应用程序SQL脚本能够高效、准确地迁移至DWS数据库，同时减少迁移过程中的风险和停机时间？DSC是一款运行在Linux或Windows操作系统上的命令行工具，致力于向客户提供简单、快速、可靠的应用程序SQL脚本迁移服务，通过内置的语法迁移逻辑解析源数据库应用程序sql脚本，并迁移为适用于DWS数据库的应用程序sql脚本。

DSC不需要连接数据库，可在离线模式下实现零停机迁移，迁移过程中还会显示迁移过程状态，并用日志记录操作过程中发生的错误，便于快速定位问题。

⚠ 注意

DSC内置语法解析器只能识别标准的SQL语法，输入脚本中如包含了其他语言的文本，则会导致语法解析异常，请在使用DSC工具前对输入脚本进行清洗。

迁移对象

DSC支持迁移的数据源如表5-1，具体支持迁移的数据库对象有：

- DDL语句：模式、表、视图、存储过程、自定义函数
- DML语句：select、update、delete、insert、truncate

表 5-1 支持的源数据库

数据库名称	数据库版本
Teradata	17.20
MySQL	8.0
Oracle	11g Release 2，12c Release 1

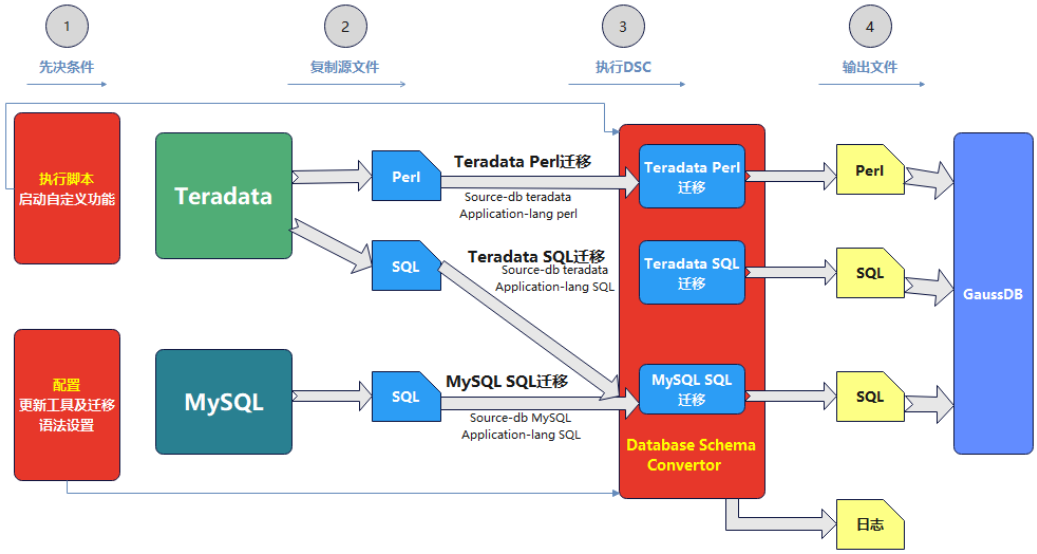
数据库名称	数据库版本
AnalyticDB For MySQL	-
BigQuery	-
Postgres	9.2.x，11.x
Redshift	-
Greenplum	7.0.0
Hologres	3.1.17
Netezza	-
Hive	-
其他源：Doris, StarRocks, SQL Server, Synapse, Impala	-

迁移流程

DSC迁移sql脚本流程如下：

1. 从源数据库导出待迁移的sql脚本到已安装了DSC的Linux或Windows服务器。
2. 执行DSC命令进行语法迁移，命令中指定输入文件路径、输出文件路径以及日志路径。
3. DSC自动将迁移后的sql脚本和日志信息归档在指定路径中。

图 5-1 DSC 处理流程



硬件要求

DSC对硬件的要求如表5-2所示。

表 5-2 DSC 硬件环境要求

硬件	配置
CPU	AMD或Intel Pentium （最小频率：500 MHz）
最小内存	1 GB
磁盘空间	1 GB

软件要求

操作系统要求

DSC兼容的操作系统如表5-3所示。

表 5-3 兼容的操作系统

服务器	操作系统	版本
通用x86服务器	SUSE Linux Enterprise Server 11	SP1（SUSE11.1）
		SP2（SUSE11.2）
		SP3（SUSE11.3）
		SP4（SUSE11.4）
	SUSE Linux Enterprise Server 12	SP0（SUSE12.0）
		SP1（SUSE12.1）
		SP2（SUSE12.2）
		SP3（SUSE12.3）
	RHEL	6.4-x86_64（RedHat6.4）
		6.5-x86_64（RedHat6.5）
		6.6-x86_64（RedHat6.6）
		6.7-x86_64（RedHat6.7）
		6.8-x86_64（RedHat6.8）
		6.9-x86_64（RedHat6.9）
		7.0-x86_64（RedHat7.0）
		7.1-x86_64（RedHat7.1）
		7.2-x86_64（RedHat7.2）
		7.3-x86_64（RedHat7.3）
		7.4-x86_64（RedHat7.4）

服务器	操作系统	版本
	CentOS	6.4（CentOS6.4）
		6.5（CentOS6.5）
		6.6（CentOS6.6）
		6.7（CentOS6.7）
		6.8（CentOS6.8）
		6.9（CentOS6.9）
		7.0（CentOS7.0）
		7.1（CentOS7.1）
		7.2（CentOS7.2）
		7.3（CentOS7.3）
		7.4（CentOS7.4）
	Windows	7.0，10，11

其他软件要求

DSC对其他软件版本的要求如表5-4所示。

表 5-4 其他软件要求

软件	用途
JDK 1.8.0_141 or later（必选）	Used to run DSC.
Perl 5.8.8（可选）	Used to migrate Perl files.
Perl 5.28.2 and later（可选）	Used to migrate Perl files in Windows.
Python 3.8.2（可选）	Used to verify post migration script.

命令行格式约定

本手册中可能出现下列命令行格式约定，它们所代表的含义如下：

表 5-5 命令行格式列表

格式	说明
粗体	命令行关键字（命令中保持不变、必须照输的部分）采用加粗字体表示。
斜体	命令行参数，路径，文件或文件夹采用斜体表示。
[ ]	表示用“[ ]”括起来的部分（关键词和参数）在命令配置时是可选的。
{ x | y | ... }	表示用“{}”分组选项，各选项之间以“|”分隔。从两个或多个选项中选取一个。
[x | y | ...]	表示用“[ ]”分组选项，各选项之间以“|”分隔。从两个或多个选项中选取一个或者不选。
{ x | y | ... }*	表示用“{}”分组选项，各选项之间以“|”分隔。从两个或多个选项中选取多个，最少选取一个，最多选取所有选项。
[x | y | ...]*	表示用“[ ]”分组选项，各选项之间以“|”分隔。从两个或多个选项中选取多个或者不选。
&<1-n>	表示&符号前的内容可重复1到n次。
#	表示注释。

第三方许可

本节包含适用于该工具的第三方许可。

- ANTLR v4.9.3
- Apache Commons IO 2.11
- Apache Commons CLI 1.5
- Apache Log4j 2.17.2
- JSON.org json 20220320
- postgresql 42.4.1
- sql-formatter 2.0.3

5.2 下载并安装 DSC

在使用DSC工具之前，必须在Linux或Windows服务器中安装工具，DSC支持Linux 64位操作系统。DSC支持其它操作系统的详情请见**了解DSC**。

前提条件

- 在Linux系统中请勿使用具有root权限的用户安装和操作DSC。且该用户必须具有创建文件夹的权限，否则install.sh将执行失败。

- 请确保目标文件夹大小至少为输入文件夹中SQL文件大小的4倍。

  例如，输入文件夹中SQL文件大小为100 KB，则目标文件夹至少需要400 KB空间处理SQL文件。

  说明

  - Linux系统中查询目标文件夹的可用磁盘空间：
    ```
    df -P <folder path>
    ```
  - Linux系统中查询输入文件的大小，在输入文件所在路径下执行：
    ```
    ls -l
    ```

- 系统已安装JRE 1.8及以上版本和Perl。有关软硬件环境的具体要求，请参见**了解DSC**。

  执行以下步骤验证Java安装版本并设置Java路径。

  a. 验证Java安装是否符合要求。
     ```
     java -version
     ```
  b. 验证java路径是否设置，如果不正确请按照步骤重新设置。

     - Linux

       1) 验证Java路径是否设置。
          ```
          echo $JAVA_HOME
          ```
       2) 如果命令返回为空，请编辑当前用户的.bashrc文件，输入如下内容，保存并退出。

          假设Java安装路径为“/home/user/Java/jdk1.8.0_141”。
          ```
          export JAVA_HOME=/home/user/Java/jdk1.8.0_141
          export PATH=$JAVA_HOME/bin:$PATH
          ```
       3) 激活Java环境变量。
          ```
          source ~/.bashrc
          ```

     - Windows

       1) 在“我的电脑”右键菜单中选择“属性”，弹出“系统”窗口。
       2) 单击“高级系统设置”，弹出“系统属性”对话框。
       3) 在“高级”页签中单击“环境变量”，弹出“环境变量”对话框。
       4) 选中“系统变量”中的“Path”环境变量，单击下方的“编辑”按钮，查看Path中是否包含Java安装路径。

          如果Path中未包含Java安装路径或路径不正确，请在原有内容的基础上增加本机的Java路径。

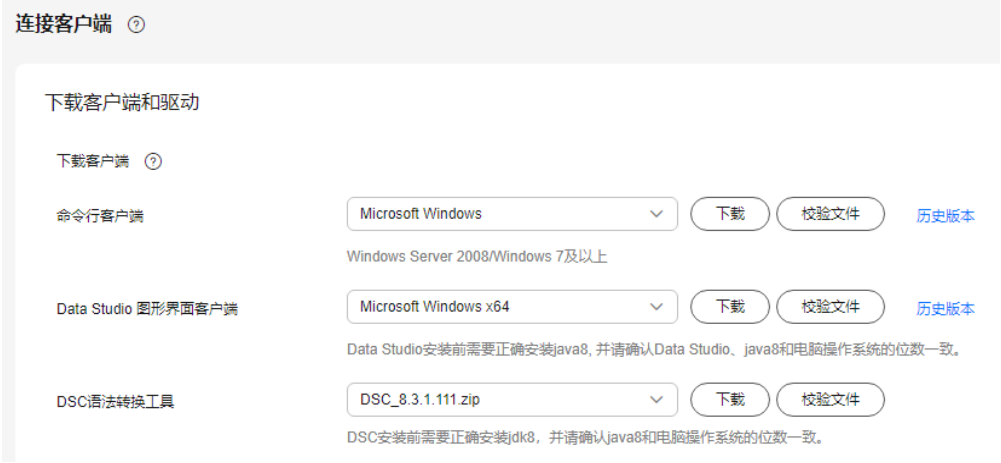
          假设Java安装路径为“C:\Program Files\Java\jdk1.8.0_141\bin”，Path环境变量为“*c:|windows|system32;*，则Path应该设置为“*c:|windows|system32;***C:\Program Files\Java\jdk1.8.0_141\bin;**”。

下载 DSC 工具-界面方式

步骤1 登录DWS管理控制台。在左侧导航栏中，单击“管理 > 连接客户端”，进入“下载客户端和驱动”页面。

步骤2 在“下载客户端和驱动”页面，单击“这里”下载“DSC”软件压缩包。

图 5-2 下载客户端



步骤3 解压下载的客户端软件包到自定义路径下。

----结束

安装 DSC 工具

DSC是一款运行在Linux或Windows操作系统上的命令行工具，可免安装使用，下载软件包后，用户解压软件包即可使用。

Windows操作系统:

步骤1 解压DSC.zip包。

得到DSC文件夹。

说明

解压DSC.zip时，可根据需要选择任意文件夹进行解压。

步骤2 进入DSC目录。

步骤3 找到并查看DSC目录中的文件。

解压出来的文件夹和文件说明如表5-6所示。

----结束

Linux操作系统:

步骤1 进入DSC.zip所在目录，解压DSC.zip文件，其中DSC.zip请以实际获取的包名为准。

```
unzip DSC.zip
```

步骤2 进入DSC目录。

```
cd DSC
```

步骤3 查看DSC目录中的文件。

```
ls
config   lib   scripts   bin   input output runDSC.sh   runDSC.bat
```

----结束

表 5-6 DSC 目录

文件或文件夹	说明
bin	dsc相关jar（可执行的）。
config	DSC工具的配置文件。
input	输入文件夹
lib	该文件夹中包括DSC正常运行所必须的库文件。
output	输出文件夹
scripts	该文件夹中包括Oracle和Teradata迁移的自定义配置脚本，用户可以直接执行sql文件启用所需功能。
runDSC.sh	在Linux操作系统中运行应用程序。
runDSC.bat	在Windows操作系统中运行应用程序。

说明

如果不再需要DSC，可以通过删除DSC文件夹本身来卸载它。

5.3 配置 DSC

5.3.1 DSC 配置

DSC的配置包含如下内容：

- **设置application.properties**：用于配置工具的迁移行为，例如，是否要覆盖目标文件夹下的文件，是否对sql文件格式化。
- **设置Java内存分配**：用户配置工具在迁移过程中可使用的内存资源，超出设置的内存，工具将显示错误消息并退出。

设置 application.properties

application.properties文件中包括一系列应用配置参数，用于控制DSC在迁移数据库脚本时的行为，该文件中的参数为通用控制参数，适用于Teradata、MySQL迁移。

设置方法如下。

步骤1 打开config文件夹中的application.properties文件。

步骤2 根据实际需要修改application.properties文件中参数的值。

application.properties文件中的参数解释见**表5-7**。

说明

- 参数值不区分大小写。
- 除了列出的参数外，用户不得更改任何参数值。

步骤3 保存后退出。

----结束

表 5-7 application.properties 文件的配置参数

参数	说明	取值范围	默认值	样例
• formatterrequired	若该参数配置发生改变，工具将无法按预期运行。	• true • false	true	formatterrequired=true
• prevalidationFlag	若该参数配置发生改变，工具将无法按预期运行。	• true • false	true	prevalidationFlag=true
• commentSeparatorFlag	若该参数配置发生改变，工具将无法按预期运行。	• true • false	true	commentSeparatorFlag=true
• queryDelimiter	若该参数配置发生改变，工具将无法按预期运行。	NA	不适用	queryDelimiter=;
• blogicDelimiter	若该参数配置发生改变，工具将无法按预期运行。	NA	不适用	blogicDelimiter=/
• Timeout	工具迁移超时时间。 若该参数配置发生改变，工具将无法按预期运行。	-	4	Timeout=4

参数	说明	取值范围	默认值	样例
• fileExtension	合法的文件扩展名，以逗号分隔。 如果此配置项fileExtension有修改，工具将不能正常运行。 **说明** 导出的脚本必须带有如下后缀，如： • .sql • .txt • .fnc • .proc • .tbl • .tbs • .pl • .dsql 等。	• csv • txt • SQL	SQL	fileExtension=SQL
• formattedSourceRequired	指定是否使用SQL Formatter对源SQL文件进行格式化。 若该参数设为true，则对输入文件的副本进行格式化并保存到*{输出路径}/formattedSource*文件夹。	• true • false	true	formattedSourceRequired=true

参数	说明	取值范围	默认值	样例
• target_files	指定要在输出/目标文件中执行的操作。 Overwrite：用于覆盖输出文件夹中的现有文件。 指定是否必须覆盖输出文件夹中的文件。 Delete：用于删除目标文件夹中的所有文件。 Cancel：用于在输出/目标文件夹中存在文件时取消操作。	• overwrite • delete • cancel	overwrite	target_files=overwrite
• encodingFormat	指定输入/源文件的编码格式。 如果未设置该参数（或该参数被注释掉），则工具将基于区域设置使用默认编码。 **说明** • 文件编码的自动检测功能并不准确。为确保正确的编码格式，请使用本参数指定格式。	• UTF8 • UTF16 • UTF32 • GB2312 • ASCII等	基于区域设置的默认编码	encodingFormat=UTF8
• NoOfThreads	指定用于迁移的线程数。	取决于可用的系统资源	3	NoOfThreads=3

参数	说明	取值范围	默认值	样例
• MaxFileSizeWarning	指定输入文件大小告警阈值，单位为B（字节）、KB、MB或GB。 如果指定的值无效，那么将使用默认值。 如果指定的源文件大小超过指定的值，则会向用户显示以下警告消息： ********** [WARNING] : Migration of the following files(>100KB) will take more time: bigfile001.SQL bigfile008.SQL **********	10 KB~1 GB	10MB	MaxFileSizeWarning=10MB
• MaxFileSize	允许输入文件的最大大小，如果超过这个限制，文件迁移将被跳过。	-	20MB	MaxFileSize=20MB

参数	说明	取值范围	默认值	样例

• MaxSqlLen	指定待迁移的单个查询的最大长度。
	如果指定的值无效，则工具会将其重置为默认值，并且控制台上将显示以下错误消息：
	`The query length parameter (MaxSqlLen) value is out of range. Resetting to default value.`
	如果输入查询超过指定的最大长度，则查询迁移会预验证失败。工具会跳过该查询，并记录以下错误消息：
	`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`
		1 .. 52,428,800 字节
		（1字节 - 50 MB）
			1048576（1 MB）	MaxSqlLen=1048576

• initialJVMMemory	设置初始内存	NA	256 MB	initialJVMMemory=256MB 表明内存达到256MB时，进程将启动。
• maxJVMMemory	设置最大内存	NA	1024 MB	maxJVMMemory=2048m 表明内存达到2048 MB时，进程将启动。

参数	说明	取值范围	默认值	样例
• executesqlingauss	在DWS中运行迁移后的脚本。参数值范围为true/false。脚本仅可在Linux系统的服务器上运行。	• true • false	false	executesqlingauss=false

说明

- 如果为配置参数提供了错误或无效值，DSC将采用该参数的默认值。
- 如果存在扩展名不受支持（如“.doc”），建议将此扩展名添加到“application.properties”文件的“fileExtension”配置参数中。

设置 Java 内存分配

DSC支持通过参数控制Java虚拟机（JVM）的内存分配量，并预设默认值。

在迁移操作期间，如果内存使用超过设置的值，DSC将提示“java.lang.OutOfMemoryError: GC overhead limit exceeded”错误消息并退出，此时用户可通过更改application.properties配置文件中的initialJVMMemory和maxJVMMemory 的值，以分配更多内存。

说明

可用系统资源决定了内存分配量。

表 5-8 JVM 内存分配的控制参数

参数	说明	推荐取值
Xms	指定初始内存分配量，单位为MB。	该参数最小值为256 MB，最大值取决于可用的系统资源。 默认值：256
Xmx	指定内存分配量的上限，单位为MB。	该参数最小值为1024 MB，最大值取决于可用的系统资源。 默认值：1024

打开校验模块config文件夹下的gaussdb.properties文件，参照**表5-9**，配置参数以连接DWS。

表 5-9 gaussdb.properties 文件内参数

参数名	描述	取值范围	默认值	样例
gaussdb-user	DWS数据库用户，拥有全部权限。	NA	NA	user1
gaussdb-port	DWS数据库端口号。	NA	NA	8000
gaussdb-name	DWS的数据库名称。	NA	NA	gaussdb
gaussdb-ip	DWS数据库IP地址。	NA	NA	10.XX.XX.XX

5.3.2 Teradata SQL 迁移

设置Teradata配置参数可在迁移Teradata数据库脚本时自定义迁移工具的行为。

打开config文件夹中的features-teradata.properties文件，并根据实际需要设置表5-10中的参数。

表 5-10 表 1 features-teradata.properties 文件中的配置参数

参数	说明	取值范围	默认值	样例
• deleteToTruncate	该参数用于设置不含WHERE的DELETE语句迁移规则。 若该参数设为true，则可将DELETE迁移为TRUNCATE。若该参数设为false，则不可将DELETE迁移为TRUNCATE。	• true • false	false	deleteToTruncate=true

参数	说明	取值范围	默认值	样例
• distributeByHash	基于主索引中指定的字段，将数据分布在集群多个节点上。 若该参数设为one，表示数据基于主索引的首个字段分布。 若该参数设为many，表示数据基于所有主索引字段分布。 该功能通过指定DISTRIBUTE BY子句实现。 **说明** 该参数在V100R002C60版本中设置为one，因为该版本不支持在DISTRIBUTE BY子句中指定多个字段。	• one • many	many	distributeByHash=many
• extendedGroupByClause	该参数用于启用和禁用Group By（grouping sets/cube/rollup）迁移。 若该参数设为true，则可迁移GROUP BY()。 若该参数设为false，则不可迁移GROUP BY()。	• true • false	false	extendedGroupByClause=false
• inToExists	该参数可用于启用和禁用从IN/NOT IN到EXISTS/NOT EXISTS的查询优化。	• true • false	false	inToExists=false
• rowstoreToColumnstore	该参数将rowstore（行存）表转换为COLUMN（列存）表。 如果该参数设为true，则所有rowstore表脚本迁移时会转换为columnstore表。	• true • false	false	rowstoreToColumnstore=false

参数	说明	取值范围	默认值	样例
• session_mode	该参数用于在运行CREATE TABLE时设置默认表类型（SET/MULTISET）。 若该参数设为Teradata，则默认表类型会配置为SET。 若该参数设为ANSI，则默认表类型会配置为MULTISET。	• Teradata • ANSI	Teradata	session_mode=ANSI
• tdMigrateALIAS	该参数用于启用/禁用ALIAS迁移。 若该参数设为true，则迁移ALIAS。 若该参数设为false，则不迁移ALIAS。	• true • false	false	tdMigrateALIAS=true
• tdMigrateDOLLAR	该参数用于设置迁移工具行为，从而迁移名称以$（美元符号）开头的静态对象。该参数不适用于动态对象，这些对象的名称使用${}格式。 若该参数设为true，则使用英文双引号（"）将以$开头的对象名称括起来。 若该参数设为false，则直接迁移以$开头的对象。	• true • false	true	tdMigrateDOLLAR=true
• tdMigrateLOCKoption	该参数是否迁移包含LOCK关键字的查询。 true表示在迁移此类查询时注释掉LOCK功能（LOCK到ACCESS）。 false表示不迁移此类查询。工具会跳过此查询，并记录以下消息： Gauss does not have equivalent syntax for LOCK option in CREATE VIEW and INSERT statement. Please enable the config_param tdMigrateLockOption to comment the LOCK syntax in the statement.	• true • false	false	tdMigrateLOCKoption=true

参数	说明	取值范围	默认值	样例
• tdMigrateNULLIFZERO	该参数指定是否迁移NULLIFZERO()。 若该参数设为true，则迁移NULLIFZERO()。 若该参数设为false，则不迁移NULLIFZERO()。	• true • false	true	tdMigrateNullIFZero=true
• tdMigrateVIEWCHECKOPTION	该参数指定是否迁移包含CHECK OPTION的视图。 若该参数设为true，则迁移时注释掉此类视图。 若该参数设为false，则不迁移此类视图。工具将按原样复制此查询并记录以下消息： Gauss does not support WITH CHECK OPTION in CREATE VIEW. Please enable the config_param tdMigrateViewCheckOption to comment the WITH CHECK OPTION syntax in the statement.	• true • false	false	tdMigrateVIEWCHECKOPTION=true
• tdMigrateZEROIFNULL	该参数指定是否迁移ZEROIFNULL。 若该参数设为true，则迁移ZEROIFNULL()。 若该参数设为false，则不迁移ZEROIFNULL()。	• true • false	true	tdMigrateZEROIFNULL=true
• volatile	特定会话的volatile数据和表仅存储在该会话中。会话结束后，其数据和表会删除。 volatile表可以是表迁移表或unlogged表。 **说明** V100R002C60仅支持unlogged表选项，不支持local temporary表。	• local temporary • unlogged	local temporary	volatile=unlogged

参数	说明	取值范围	默认值	样例
• tdMigrateCharsetCase	该参数指定是否迁移CHARACTER SET和CASESPECIFIC。 若该参数设为true，则迁移CHARACTER SET和CASESPECIFIC为注释掉的脚本。 若该参数设为false，则不迁移CHARACTER SET和CASESPECIFIC。工具按原样复制查询，并在错误日志文件中记录以下消息，包括查询细节（如文件名和语句位置）： CHARACTER SET和CASESPECIFIC是列级选项，Gauss不提供等效语法。 用户可以改写相应语句，或将tdMigrateCharsetCase参数设为true，从而注释掉CHARACTER SET和CASESPECIFIC。	• true • false	false	tdMigrateCharsetCase=false **说明** 如果tdminatecharsetcase = true，则注释该字符的特殊关键字。
• terdataUtilities	是否支持迁移Teradata命令行工具。 支持以下参数值： • true • false	• true • false	true	terdataUtilities=true
• unique_primary_index_in_column_table	是否支持为列存表创建unique索引。	• true • false	true	unique_primary_index_in_column_table=true
• default_charset	是否支持迁移default_charset。 支持以下参数值： • LATIN • UNICODE • GRAPHIC	• LATIN • UNICODE • GRAPHIC	LATIN	default_charset=LATIN

参数	说明	取值范围	默认值	样例
• mergeImplementation	指定merge类型： • 使用WITH子句 • 拆分查询 支持以下参数值： • With • Split • None	• With • Split • None	None	mergeImplementation=None
• dsqlSupport	是否支持dsql。 支持以下参数值： • true • false	• true • false	false	dsqlSupport=false
• tdcolumnInSensitive	是否在迁移时删除包含双引号的列名称。 支持以下参数值： • true • false	• true • false	false	tdcolumnInSensitive=false
• tdMigrateCASE_N	指定分区关键字CASE_N的迁移方式。Gauss不支持多级（嵌套）分区。 支持以下参数值： • comment • none	• comment • none	comment	tdMigrateCASE_N=comment
• tdMigrateRANGE_N	指定分区关键字RANGE_N的迁移方式。Gauss不支持多级（嵌套）分区。 支持以下参数值： • comment • none • range	• comment • none • range	range	tdMigrateRANGE_N=range

参数	说明	取值范围	默认值	样例
• tdMigrateAddMonth	是否支持迁移addMonth。 支持以下参数值： • true • false 若该参数设为true，则迁移后为mig_td_ext.ADD_MONTHS（添加mig_td_ext）。否则，不支持迁移。	• true • false	false	tdMigrateAddMonth=false

5.3.3 Teradata Perl 迁移

设置Teradata Perl配置参数可在迁移Teradata Perl数据库脚本时自定义迁移工具的行为。

打开config文件夹中的perl-migration.properties文件，并根据实际需要设置表5-11中的参数。

说明

- 参数值不区分大小写。
- 用户可以更改下表中db-bteq-tag-name和db-tdsql-tag-name的参数值。

表 5-11 表 1 perl-migration.properties 文件中的配置参数

参数	说明	取值范围	默认值	样例
• db-bteq-tag-name	指定要在Perl文件中处理的脚本。 BTEQ：仅处理BTEQ标记的脚本。	• bteq	bteq	db-bteq-tag-name=bteq
• db-tdsql-tag-name	指定待处理的脚本。 SQL_LANG：仅处理SQL_LANG标记的脚本。	sql_lang	sql_lang	db-tdsql-tag-name=sql_lang

参数	说明	取值范围	默认值	样例
• add-timing-on	指定是否通过添加脚本来计算执行时间。 如果启用，则为每个输入文件添加add timing脚本。	• true • false	false	add-timing-on=true
• remove-intermediate-files	指定在迁移完成后是否删除工具创建的中间SQL文件。 该文件包含SQL文件中的BTEQ和SQL_LANG语法，作为语法迁移工具的输入文件。 设为true，表示删除中间文件。 设为false，表示不删除中间文件。	• true • false	true	remove-intermediate-files=true

参数	说明	取值范围	默认值	样例

• migrate-variables	指定是否迁移分配给Perl变量的SQL命令。
	Perl文件可以包含分配了SQL命令的Perl变量，通过PREPARE和EXECUTE语句在Perl中执行。该工具还可以从Perl变量提取SQL命令进行迁移。
	设为true，表示对分配给Perl变量的SQL命令进行迁移。
	设为false，表示在迁移时跳过该Perl变量。
	示例1：
	当参数设为true时，将
	`$V_SQL = "CT X1(C1 INT,C2 CHAR(30))";`
	迁移为：
	`CREATE TABLE X1(C1 INT,C2 CHAR(30));`
	示例2：
	输入
	`$onesql ="SELECT trim(tablename) from dbc.tables WHERE databasename = '${AUTO_DQDB}' and tablename like 'V_%' order by 1;"; $sth_rundq = $dbh->execute_query($onesql);`
	输出
	`$onesql ="SELECT TRIM( tablename ) FROM dbc.tables WHERE databasename = '${AUTO_DQDB}' AND tablename LIKE 'V_%' ORDER BY 1 ;"; $sth_rundq =`
		• true
		• false
			true	migrate-variables=true

参数	说明	取值范围	默认值	样例
	`$dbh->execute_query($one sql);`			
• logging-level	指定Perl迁移日志的级别。 设为error，表示仅记录错误日志。 设为warning，表示记录错误及告警日志。 设为info，表示记录错误、告警和活动日志。该级别包含所有日志信息。	• error • warning • info	info	logging-level=info
• log-file-count	指定保留日志文件的最大数量。文件总数包括正在使用的日志文件和已归档的日志文件。 如果新归档的日志文件超过了文件数上限，则会先删除最早保留的文件，直到成功保存指定数量的文件。	3 - 10	5	log-file-count=10
• log-file-size	指定日志文件的最高上限。 日志文件大小达到指定上限时，给文件名添加时间戳并进行归档。 示例：`perlDSC_2018-07-08_16_12_08.log` 归档后，生成新的日志文件perlDSC.log。	1MB - 10MB	5MB	log-file-size=10MB

参数	说明	取值范围	默认值	样例

• migrate-executequery	指定是否迁移包含SQL内容的execute_query。
	设为true，表示迁移。
	设为false，表示不迁移。
	示例：
	设为true时，将
	`my $rows1=$conn1->execute_query("sel ${selectclause} from ${databasename}.${tablename};");`
	迁移为：
	`my $rows1=$conn1->execute_query("SELECT ${selectclause} FROM ${databasename}.${tablename} ;");`
		• true
		• false
			true	migrate-executequery=true

5.3.4 MySQL（BigQuery、ADB for MySQL、SQL-Server）配置

设置MySQL配置参数可在迁移MySQL、BigQuery、ADB for MySQL、SQL-Server数据库脚本时自定义迁移工具的行为。

打开config文件夹中的features-mysql.properties文件，并根据实际需要设置**features-mysql.properties文件中的配置参数**中的参数。

表 5-12 features-mysql.properties 文件中的配置参数

参数	说明	取值范围	默认值	样例
• table.databaseAsSchema • table.defaultSchema	是否使用数据库名称作为schema名称，如果数据库名称不存在，则使用用户定义schema，如果用户定义schema为空，则使用默认schema。	• true • false • public	• true • public	• table.databaseAsSchema=true • table.defaultSchema=public

参数	说明	取值范围	默认值	样例
• table.schema	用户设置的schema名称，如果该参数不为空，则使用该参数，即使包含useDatabaseAsSchema = true，也会使用当前schema名称。	• schemaName	• 默认为空	• table.schema=
• table.orientation	默认数据存储方式，ROW：行存储，COLUMN：列存储。	• ROW • COLUMN	• ROW	• table.orientation=ROW
• table.type	默认的表类型，分区表、复制表、round-robin表。REPLICATION、HASH、ROUND-ROBIN。	• HASH • REPLICATION • ROUND-ROBIN	• HASH	• table.type=HASH
• table.tablespace	表空间选项。	• COMMENT • RESERVE	• RESERVE	• table.tablespace=RESERVE
• table.partition-key.choose.strategy	分区键选择策略。	• partitionKeyChooserStrategy	• partitionKeyChooserStrategy	• table.partition-key.choose.strategy=partitionKeyChooserStrategy
• table.partition-key.name	分区键设置，如果为空，则按照默认策略选择，如果有多列，用逗号分隔，忽略列名称大小写。	• 预留参数	• 默认为空	• table.partition-key.name=

参数	说明	取值范围	默认值	样例
• table.compress.mode	创建新表时，需要在CREATE TABLE语句中指定关键字COMPRESS，这样，当对该表进行批量插入时就会触发压缩特性。该特性会在页范围内扫描所有元组数据，生成字典、压缩元组数据并进行存储。指定关键字NOCOMPRESS则不对表进行压缩。	• COMPRESS • NOCOMPRESS	• NOCOMPRESS	• table.compress.mode=NOCOMPRESS
• table.compress.row • table.compress.column	指定表数据的压缩级别，它决定了表数据的压缩比以及压缩时间。	• YES • NO • YES • NO • LOW • MIDDLE • HIGH	• NO • LOW	• table.compress.row=NO • table.compress.column=LOW
• table.compress.level	指定表数据同一压缩级别下的不同压缩水平，它决定了同一压缩级别下表数据的压缩比以及压缩时间。对同一压缩级别进行了更加详细的划分，为用户选择压缩比和压缩时间提供了更多的空间。总体来讲，此值越大，表示同一压缩级别下压缩比越大，压缩时间越长；反之亦然。	• 0 • 1 • 2 • 3	• 0	• table.compress.level=0
• table.database.template	数据库模板。	• 预留参数	• template0	table.database.template=template0

参数	说明	取值范围	默认值	样例
• table.database.encoding	A database code.	• UTF8 • SQL_ASCII • GBK • Latin1 codes	• UTF8	table.database.encoding=UTF8
• table.index.rename	创建索引时，是否重新命名索引名。	• true • false	• false	table.index.rename=false
• table.database.onlyFullGroupBy	select后非聚合列是否全部出现在group by中。	• true • false	• true	table.database.onlyFullGroupBy=true
• table.database.realAsFloat	REAL数据类型转换使用，默认false，转换为DOUBLE PRECISION；改为true时，转换为REAL。	• true • false	• false	table.database.realAsFloat=false
• table.database.havingAddGroupBy	having clause语句是否添加GROUP BY。	• true • false	• true	table.database.havingAddGroupBy=true
• table.function.uuid	uuid函数是否以原语句函数原样输出： • false：转为sys_uuid函数。 • true：输出uuid函数。 默认false。	• true • false	• false	table.function.uuid=false
• table.function.split	split 拆分函数： • 默认true：输出split函数。 • false：转为split_part函数。	• true • false	• true	table.function.split=true

参数	说明	取值范围	默认值	样例
• table.function.tryCast	try_cast函数是否以原语句函数原样输出。 • 默认false：转为cast函数。 • true：输出try_cast函数。	• true • false	• false	table.function.tryCast=false
• table.sql.set.names	setNames语句是否注释，不转换。 • 默认true：注释，不转换。 • false：进行转换。	• true • false	• true	table.sql.set.names=true
• table.sql.set.password	setPassword语句是否注释，不转换。 • 默认true：注释，不转换。 • false：进行转换。	• true • false	• true	table.sql.set.password=true
• table.sql.set.resource.group	setResourceGroup语句是否注释，不转换。 • 默认true：注释，不转换。 • false：进行转换。	• true • false	• true	table.sql.set.resource.group=true
• table.sql.set.role	setRole语句是否注释，不转换。 • 默认true：注释，不转换。 • false：进行转换。	• true • false	• true	table.sql.set.role=true
• table.sql.set.variable	setVariable语句是否注释，不转换。 • 默认true：注释，不转换。 • false：进行转换。	• true • false	• true	table.sql.set.variable=true

参数	说明	取值范围	默认值	样例
• sql.conversion.type	SQL文件的转换类型:ddl/plsql。 默认为ddl，当要执行存储过程或自定义函数时转换类型为plsql。	• ddl • plsql	• ddl	sql.conversion.type=ddl
• table.insert.convert.upsert	插入时主键或唯一索引冲突的处理方式： • true (更新)：当发生冲突时，用新数据更新已存在的记录。 • false (忽略)：当发生冲突时，放弃新数据，保留原有的记录。	• true • false	• false	table.insert.convert.upsert=false
• table.alter.add.index	是否支持alter table ... add index转换。 默认false不转换, true转换。	• true • false	• false	table.alter.add.index=false
• table.create.comment.constraint	是否注释多余的约束， 默认false不注释。	• true • false	• false	table.create.comment.constraint=false
• table.create.type.set	是否对set类型增加check约束，默认false不增加。	• true • false	• false	table.create.type.set=false
• table.create.type.enum	是否对enum类型增加 in not null约束，默认false不增加。	• true • false	• false	table.create.type.enum=false
• table.create.columnCase	是否对列字段进行小写转换，默认为true，进行小写转换；false为与原sql保持大小写一致。	• true • false	• true	table.create.columnCase=true
• table.option.colversion	指定列存存储格式的版本，默认值2.0，可选值 1.0、2.0、3.0。	• 1.0 • 2.0 • 3.0	• 2.0	table.option.colversion=2.0

参数	说明	取值范围	默认值	样例
• table.origin.database.type	源数据库类型，包括bigquery、doris、 synapse、mysql&adb，默认为mysql&adb。	• bigquery • doris • synapse • mysql • adb	• mysql&adb	table.origin.database.type = mysql
• table.enable.hstore	是否允许hstore表，true/false(默认为空，false代表在语句中设置hstore为false)。	• true • false • 空	• 默认为空	table.enable.hstore=
• table.enable.hstore.opt	是否允许hstore_opt操作，true表示运行，false表示禁止。	• true • false	• true	table.enable.hstore.opt=true
• table.partition.comment.enable	是否将partition的原始SQL输出为注释，true/false。	• true • false	• false	table.partition.comment.enable=false
• table.type.schema.prefix	创建type时，type名称是否拼接schema名，true/false。	• true • false	• true	table.type.schema.prefix=true
• table.option.partition.ttl	分区参数partition_expiration_days需要转为整数，指定四舍五入规则。 • HALF_UP：四舍五入。 • CEILING：向上取整。 • FLOOR：向下取整。	• HALF_UP • CEILING • FLOOR	• HALF_UP	table.option.partition.ttl=HALF_UP

5.3.5 Oracle 配置

设置Oracle配置参数可在迁移Oracle数据库脚本时自定义迁移工具的行为。

打开config文件夹中的antlr-features-oracle.properties文件，并根据实际需要设置**表1 antlr-features-oracle.properties文件中的配置参数**中的参数。

表 5-13 antlr-features-oracle.properties 文件中的配置参数

参数	说明	取值范围	默认值	样例
• table.index.rename	操作索引时，是否重新命名索引名"表名_索引名"。	• true • false	• false	table.index.rename=false
• table.constraint.rename	操作约束时，是否重新命名约束名"表名_约束名"	• true • false	• false	table.constraint.rename=false
• table.foreign.key.comment	是否注释外键约束	• true • false	• true	table.foreign.key.comment=true
• table.storage.para.comment	存储参数是否注释	• true • false	• true	table.storage.para.comment=true
• table.plsql.exception	是否注释PL/SQL中的异常模块,默认值为true,注释异常模块	• true • false	• true	table.plsql.exception=true;
• table.create.inSensitive	是否对列字段进行小写转换，默认为true，进行小写转换，false为与原sql保持大小写一致	• true • false	• true	table.create.inSensitive=true
• table.plsql.pkgConvert	当前是否进行包替换，默认为false，进行普通转换，true进行包变量替换	• true • false	• false	table.plsql.pkgConvert=false

5.3.6 PostgreSql 配置

设置PostgreSql配置参数可在迁移PostgreSql、Greenplum、Netezza数据库脚本时自定义迁移工具的行为。

打开config文件夹中的features-pg.properties文件，并根据实际需要设置表5-14中的参数。

表 5-14 features-pg.properties 文件中的配置参数

参数	说明	取值范围	默认值	样例
• table.orientation	默认数据存储方式，ROW：行存储，COLUMN：列存储。	• COLUMN • ROW	• COLUMN	• table.orientation=COLUMN
• table.type	默认的表类型，分区表、复制表、round-robin表，参数取值为：REPLICATION, HASH, ROUND-ROBIN。	• REPLICATION • HASH • ROUND-ROBIN • 空	• 默认为空	• table.type=
• table.compress.level	指定表数据的压缩级别，它决定了表数据的压缩比以及压缩时间。 • 行存表的有效值为YES/NO • 列存表的有效值为YES/NO/LOW/MIDDLE/HIGH 默认值为空，不设置。	行存表： • YES • NO 列存表： • YES • NO • LOW • MIDDLE • HIGH	• 默认为空	• table.compress.level=
• table.colversion.version	指定列存存储格式的版本，支持不同存储格式版本之间的切换。取值范围：空、1.0或2.0，默认为空，不指定。	• 1.0 • 2.0 • 空	• 默认为空	• table.colversion.version=
• table.enable.delta	指定了在列存表是否开启delta表，该参数只对列存表有效。取值：true、false，默认值为空。	• true • false • 空	• 默认为空	• table.enable.delta=
• table.output.placeholder.format	参数通过 String.format 模板（如 $%s）定义变量输出格式，其中 %s 会被实际变量名替换。	• StringFormat	• 默认为空	table.output.placeholder.format=

参数	说明	取值范围	默认值	样例
• table.output.foreign.server	GDS外表默认server。	• serverName	• gsmpp_server	• table.output.foreign.server=gsmpp_server
• table.output.foreign.obs.server	OBS外表默认server。	• serverName	• obs_server	• table.output.foreign.obs.server=obs_server
• table.output.char.coefficient	char长度系数（数字），char字段存储逻辑不一致时使用，char长度会乘上此系数	• 整数	• 1	• table.output.char.coefficient=1
• table.output.varchar.coefficient	varchar长度系数（数字），varchar字段存储逻辑不一致时使用，varchar长度会乘上此系数	• 整数	• 1	• table.output.varchar.coefficient=1
• table.output.bpchar.coefficient	bpchar长度系数（数字），bpchar字段存储逻辑不一致时使用，bpchar长度会乘上此系数	• 整数	• 1	• table.output.bpchar.coefficient=1
• table.output.generate.sequence	当create table语句包含nextval时，是否自动生成create sequence语句(取值Y/N)	• Y • N	• Y	• table.output.generate.sequence=Y
• table.output.column.keyword	以后续跟着的关键字作为列名时，会自动将列名拼接上``符号。 例如 SELECT `group`, `performance`, `hash`, `hot`, `matched`, `reject`, `interval`, `modify`, `timestamp`	• keyWords	• 默认为已识别关键词	• table.output.column.keyword=group,performance,hash,hot,matched,reject,interval,modify,timestamp

参数	说明	取值范围	默认值	样例
• table.output.column.type.convert	用户自定义类型映射，不区分大小写，填写示例：char|varchar,character|varchar,bpchar|varchar	• oldDataType|newDataType	• 默认为空	• table.output.column.type.convert=
• table.output.partition.valueslessthan	分区方式start end转换为values less than (取值Y/N)	• Y • N	• Y	• table.output.partition.valueslessthan=Y
• table.output.role.name.convert	用户自定义角色名映射，不区分大小写，填写示例：gpadmin|dbadmin,admin|dbadmin	• oldName|newName • 空	• 默认为空	• table.output.role.name.convert=gpadmin|dbadmin

5.3.7 Hive 配置

设置HiveSQL配置参数可在迁移HiveSQL数据库脚本时自定义迁移工具的行为。

打开config文件夹中的features-hive.properties文件，并根据实际需要设置表5-15中的参数。

表 5-15 features-hive.properties 文件中的配置参数

参数	说明	取值范围	默认值	样例
• table.orientation	行列存参数，默认不指定。	• COLUMN • ROW • 空	• 默认为空	• table.orientation=
• table.distribute-key.choose.strategy	分布键选择策略，可选值为ROUNDROBIN，HASH，REPLICATION。	• ROUNDROBIN • HASH • REPLICATION • 空	• 默认为空	• table.distribute-key.choose.strategy=
• table.function.current_date	是否转换成current_date参数，默认不转换。	• true • false	• false	• table.function.current_date=false

参数	说明	取值范围	默认值	样例
• table.function.unix_timestamp	unix_timestamp是否增加一层to_timestamp嵌套，默认不添加。	• true • false	• false	• table.function.unix_timestamp=false
• table.option.doubleQuoteForAs	as别名后是否去掉或增加双引号。 • 默认值ori：表示as别名后保持原状。 • remove：统一去掉双引号。 • add：统一增加双引号。	• ori • remove • add	• ori	• table.option.doubleQuoteForAs=ori
• table.option.joinOn	配置控制 JOIN 操作的模式： • true：自动添加 ON 1=1，强制笛卡尔积关联 • false：必须明确指定关联条件（默认值）	• true • false	• false	• table.option.joinOn=false
• table.enable.hstore	是否允许hstore表。	• true • false	• false	• table.enable.hstore=false
• table.enable.hstore.opt	是否允许hstore_opt操作。	• true • false	• false	• table.enable.hstore.opt=false
• table.column.addQuote	是否将双引号的查询列统一去掉双引号并加上单引号。	• true • false	• false	• table.column.addQuote=false
• table.special.add.primary.key	定制化需求，是否给列名为id或列名为tableName_id的列增加主键，默认为false不增加，true为增加。	• true • false	• false	• table.special.add.primary.key=false

5.4 执行 DSC 进行语法迁移

本章节指导使用DSC进行语法迁移。

注意事项

- 启动迁移程序前，可以自定义指定输出文件夹路径。输入文件夹路径、输出文件夹路径以及日志路径以空格隔开。默认在工具根目录下。输入文件夹路径不能包含空格。路径空格会导致DSC执行错误。详情请参见**故障处理**。
- 如果输出文件夹中包含子文件夹或文件，DSC会在执行迁移前将其删除或者根据用户设置（config文件夹中application.properties配置文件）将其覆盖。已删除或覆盖的子文件夹或文件无法通过DSC恢复。
- 如果在同一台服务器上并发进行迁移（由同一个或不同DSC执行），不同的迁移任务必须使用不同的输出文件夹路径和日志路径。
- 用户可以通过可选参数指定日志存储路径。如果路径未指定，DSC在TOOL_HOME下自动创建log文件夹。详情请参见**DSC日志参考**。
- 单条SQL大小约束为20KB，超过此大小可能会导致执行过慢，从而转换失败。
- 为确保安全性，DSC会对其创建的文件和文件夹进行访问控制。要访问这些文件和文件夹，用户必须拥有所需权限。例如，用户需要权限600/400访问目标文件和日志文件，需要权限700访问目标文件夹和日志文件夹。此外，该工具不在日志中保存敏感数据，以确保数据安全。
- --input-folder中指定的文件或文件夹不得具有GROUP和OTHERS的写权限。 出于安全考虑，如果输入文件/文件夹具有写入权限，则该工具不会执行。
- 不得使用拥有root权限的用户在Linux中安装和执行DSC。
- DSC.jar文件中提供的umask值是系统设置值，与文件权限相关。建议用户不要修改此值。修改此值将影响文件权限。

说明

DSC是一个单机应用程序，无需与任何网络或数据库连接即可运行。它可以在与任何网络隔离的任何机器上运行。

准备工作

在迁移之前必须先创建输入文件夹和输出文件夹，并将待迁移的所有SQL脚本复制到输入文件夹中。Linux系统操作如下：

步骤1 创建输入和输出文件夹。您可以根据用户的首选项在任意位置创建文件夹。用户也可以使用默认的文件夹作为输入、输出，作为包的一部分提供。

```
mkdir input
mkdir output
```

危险

由于DSC批量无序地读取输出文件夹，因此，建议在迁移开始后不要对输入文件夹和文件进行任何修改，这些异常操作将影响DSC的输出结果。

步骤2 将所有待迁移的 SQL文件复制到输入文件夹。

说明

- 如果源文件的编码格式不是UTF-8，请执行以下步骤：
  1. 打开config文件夹中的application.properties文件。
  2. 将application.properties文件中encodingFormat参数值修改为所需的文件编码格式。
  DSC支持UTF-8、ASCII以及GB2312编码格式。encodingFormat的值不区分大小写。
- 如果需要获取Linux系统中源文件的编码格式，请在源文件所在服务器上执行以下命令：
  file -bi *<Input file name>*

----结束

环境搭建及恢复（数据库及数据库用户）

创建DWS数据库和schema

步骤1 登录Postgres系统。

```
gsql -p <port> -d postgres
drop database <database name>;
create database <database name>;
\c <database name>
GRANT ALL PRIVILEGES ON DATABASE <database name> TO <user>;
grant database to <user>;\q
gsql -p <port> -d <database name> -U <user> -W <password> -h <IP> -f
drop database <database name>;
create database <database name>;
\c <database name>;
GRANT ALL PRIVILEGES ON DATABASE <database name> TO <user>;
gsql -p <port> -d <database name> -U <user> -W <password>  -f
```

步骤2 运行Setup目录下的所有文件。

----结束

命令：

```
sh runDSC.sh -S oracle -M blogic -I <input path>
sh runDSC.sh -I input/ -S oracle -M ddl -L log_temp -P input/bulk/1_table/
```

迁移方法

用户可在Windows和Linux操作系统中执行**runDSC.sh**或**runDSC.bat**命令进行迁移，各迁移场景的命令详见**表5-16**，带“[]”内的参数为可选参数。

表 5-16 Windows 和 Linux 场景迁移

迁移场景	命令行参数
Linux场景迁移	**> ./runDSC.sh** **--source-db** *Teradata* [**--input-folder** *<input-script-path>*] [**--output-folder** *<output-script-path>*] [**--log-folder** *<log-path>*]
Windows场景迁移	**> runDSC.bat** **--source-db** *Teradata* [**--input-folder** *<input-script-path>*] [**--output-folder** *<output-script-path>*] [**--log-folder** *<log-path>*]

说明

- 命令行参数说明：
  - 简写

    --source-db 缩写为 -S。

    --input-folder 缩写为 -I。

    --output-folder 缩写为 -O。

    --log-folder 缩写为 -L。
  - source-db指定源数据库，可选值为：

    mysql

    bigquery

    teradata

    greenplum

    netezza

    oracle

    postgres
- 命令回显说明：

  Migration process start time和Migration process end time分别表示迁移开始时间和结束时间。Total process time表示迁移总时长，单位为ms。此外，迁移文件总数、处理器总数、已使用处理器数量、日志文件路径以及错误日志文件路径也会显示在控制台上。

任务示例

- 示例：将Teradata数据库的SQL文件迁移到适用于Linux系统下的DWS的SQL脚本中。

  ./runDSC.sh --source-db *teradata*
  或
  ./runDSC.sh -S *teradata*
  ./runDSC.sh --source-db *teradata* **--input-folder** D:\test\conversion\input **--output-folder** *D:|test|conversion|output* **--log-folder** *D:|test|conversion|log*
- 示例：执行以下命令，将Teradata数据库的SQL文件迁移到适用于Windows操作系统下的DWS的SQL脚本中。

  runDSC.bat --source-db *teradata*
  或
  runDSC.bat -S *teradata*

  runDSC.bat --source-db *teradata* **--input-folder** D:\test\conversion\input **--output-folder** *D:|test|conversion|output* **--log-folder** *D:|test|conversion|log*

控制台上显示迁移详情（包括进度和完成状态）：

```
********** Schema Conversion Started **********
DSC process start time : Mon Jan 20 17:24:49 IST 2020
Statement count progress 100% completed [FILE(1/1)]
Schema Conversion Progress 100% completed
**********
Total number of files in input folder : 1
Total number of valid files in input folder : 1
**********
Log file path :...../DSC/DSC/log/dsc.log
Error Log file :
DSC process end time : Mon Jan 20 17:24:49 IST 2020
DSC total process time : 0 seconds
********** Schema Conversion Completed **********
```

查看并验证输出文件

迁移流程结束后，用户可使用对比工具（例如BeyondCompare®）将输入文件与输出文件进行比较。为了简化对比过程，也可以先对源SQL文件进行格式化。

1. 在Linux操作系统上运行以下命令以查看输出文件夹中的迁移文件。Windows操作系统不再赘述。

   ```
   cd OUTPUT
   ls
   ```

   显示类似以下信息：

   ```
   formattedSource    output
   user1@node79:~/Documentation/DSC/OUTPUT> cd output
   user1@node79:~/Documentation/DSC/OUTPUT/output> ls
   in_index.sql    input.sql    Input_table.sql    in_view.sql    MetadataInput.sql
   user1@node79:~/Documentation/DSC/OUTPUT/output>
   ```

2. 使用对比工具比较输入文件和输出文件，查看迁移后SQL文件的关键字是否符合目标数据库的要求。如果不符合，请联系技术支持处理。

查看日志文件

所有执行及错误信息都会写入对应的日志文件。详情请参见**DSC日志参考**。

检查日志文件是否记录错误信息。如是，请参考**故障处理**。

5.5 DSC 常用命令

5.5.1 Version 命令迁移

功能

Version命令用于显示DSC版本号。

命令格式

Linux:

```
./runDSC.sh --version
```

Windows:

```
runDSC.bat --version
```

使用指南

Linux:

```
./runDSC.sh --version
```

Windows:

```
runDSC.bat --version
```

系统回显

```
Version: DSC (Gauss Tools v2.0.0)
```

5.5.2 Help 命令迁移

功能

help命令用于提供DSC支持的命令相关的帮助信息。

命令格式

Linux操作系统:

```
./runDSC.sh --help
```

Windows操作系统:

```
runDSC.bat --help
```

命令示例

Linux操作系统:

```
./runDSC.sh --help
```

Windows操作系统:

```
runDSC.bat --help
```

系统回显

Linux操作系统:

```
./runDSC.sh --help
To migrate teradata/oracle/netezza/mysql/db2 database scripts to DWS
runDSC.sh -S <source-database> [-T <target-database>] -I <input-script-path> -O <output-script-path> [-M <conversion-type>]  [-A <application-lang>]  [-L <log-path>]  [-VN <version-number>]


  -S | --source-db
  The source database, which can be either Teradata or Oracle or Netezza or MySql or DB2.


  -T | --target-db
  The target database, which can be either GaussDBT or GaussDBA.


  -I | --input-folder
  The input/source folder that contains the Teradata/Oracle/Netezza/MySql/DB2 scripts to be migrated.


  -O | --output-folder
  The output/target folder where the migrated scripts are placed.


  -M | --conversion-type
  The conversion type, which can be either Bulk or BLogic.


  -A | --application-lang
  The application language type, which can be either SQL or Perl.


  -L | --log-folder
  The log file path where the log files are created.
```

```
   -VN | --version-number
   The version number, which can be either V1R7 or V1R8_330.

  " -P | --pre-execution-path%n" + "
   Path containig the dependent Objects scripts which needs to be executed before the verification step.%n"
+ "%n" +



To display DSC version details
runDSC.sh -V | --version


To display DSC help details
runDSC.sh -H | --help


[Internal] To guess encoding for a file
runDSC.sh guessencoding -F <filename>


   -F | --file-name
   The filename for which the encoding must be guessed.


Refer the user manual for more details.
```

Windows操作系统:

```
runDSC.bat --help
To migrate teradata/oracle/netezza/mysql/db2 database scripts to DWS
runDSC.bat -S <source-database> [-T <target-database>] -I <input-script-path> -O <output-script-path> [-
M <conversion-type>]  [-A <application-lang>]  [-L <log-path>]  [-VN <version-number>]


   -S | --source-db
   The source database, which can be either Teradata or Oracle or Netezza or MySql or DB2.


   -T | --target-db
   The target database, which can be either GaussDBT or GaussDBA.


   -I | --input-folder
   The input/source folder that contains the Teradata/Oracle/Netezza/MySql/DB2 scripts to be migrated.


   -O | --output-folder
   The output/target folder where the migrated scripts are placed.


   -M | --conversion-type
   The conversion type, which can be either Bulk or BLogic.


   -A | --application-lang
   The application language type, which can be either SQL or Perl.


   -L | --log-folder
   The log file path where the log files are created.


   -VN | --version-number
   The version number, which can be either V1R7 or V1R8_330.


To display DSC version details
runDSC.sh -V | --version
```

```
To display DSC help details
runDSC.sh -H | --help


[Internal] To guess encoding for a file
runDSC.sh guessencoding -F <filename>


  -F | --file-name
  The filename for which the encoding must be guessed.


Refer the user manual for more details.
```

5.6 DSC 日志参考

5.6.1 日志概述

日志文件是DSC所有操作和状态的存储库。支持以下日志文件：

表 5-17 日志文件列表

日志类别	文件名	详细说明
SQL 迁移日志	*DSC.log*	DSC.log记录 SQL 迁移过程中的所有活动。
	DSCError.log	记录SQL迁移过程中发生的错误信息。
	successRead.log	记录SQL迁移中对输入文件的成功读取次数。
	successWrite.log	记录SQL迁移中对输入文件的成功写入次数。
Perl 迁移日志	*perlDSC.log*	记录 Perl 迁移中的所有活动、警告（Warning）及错误信息。

Apache Log4j用于指定DSC记录日志的框架。用户可使用以下**表5-18**，也可以根据需要进行自定义：

表 5-18 Log4j 配置文件

适用数据库类型	配置文件路径	备注
Teradata / Oracle / Netezza / DB2	config/log4j2.xml	通用数据库配置。
MySQL	config/log4j2_mysql.xml	MySQL 专用配置。

5.6.2 SQL 迁移日志

SQL DSC（DSC.jar）支持以下类型的日志记录：

- 活动日志
- 错误日志
- 成功读
- 成功写

说明

- 如果用户指定了日志路径，所有日志都会保存在该路径下。
- 如果未指定日志路径，DSC会在TOOL_HOME路径下创建log文件夹，用于存储所有日志。
- 为控制磁盘空间用量，日志文件的大小上限为10 MB。用户最多可拥有10个日志文件。
- 工具日志不记录敏感数据，如查询。

活动日志

DSC将所有日志和错误信息保存到DSC.log文件中。该文件位于log文件夹中。DSC.log文件包含执行迁移的用户、迁移的文件、时间戳等详细信息。活动日志的记录级别为INFO。

DSC.log的文件结构如下：

```
2020-01-22 09:35:10,769 INFO CLMigrationUtility:159 DSC is initiated by xxxxx
2020-01-22 09:35:10,828 INFO CLMigrationUtility:456 Successfully changed permission of files in D:\Migration\Gauss_Tools_18_Migration\code\migration\config
2020-01-22 09:35:10,832 INFO PropertyLoader:90 Successfully loaded Property file : D:\Migration \Gauss_Tools_18_Migration\code\migration\config\application.properties
2020-01-22 09:35:10,833 INFO ApplicationPropertyLoader:42 Application properties have been loaded Successfully
2020-01-22 09:35:10,917 INFO MigrationValidatorService:549 Files in output directory has been overwritten as configured by xxxxx
2020-01-22 09:35:10,920 INFO PropertyLoader:90 Successfully loaded Property file : D:\Migration \Gauss_Tools_18_Migration\code\migration\config\features-oracle.properties
2020-01-22 09:35:10,921 INFO FeatureLoader:41 Features have been loaded Successfully
2020-01-22 09:35:10,926 INFO MigrationService:80 DSC process start time : Wed Jan 22 09:35:10 GMT +05:30 2020
2020-01-22 09:35:10,933 INFO FileHandler:179  File is not supported. D:\Migration_Output\Source \ARRYTYPE.sql-
2020-01-22 09:35:10,934 INFO FileHandler:179  File is not supported. D:\Migration_Output\Source \varray.sql-
2020-01-22 09:35:12,816 INFO PropertyLoader:90 Successfully loaded Property file : D:\Migration \Gauss_Tools_18_Migration\code\migration\config\global-temp-tables.properties
2020-01-22 09:35:12,830 INFO PropertyLoader:90 Successfully loaded Property file : D:\Migration \Gauss_Tools_18_Migration\code\migration\config\create-types-UDT.properties
2020-01-22 09:35:12,834 INFO PropertyLoader:90 Successfully loaded Property file : D:\Migration \Gauss_Tools_18_Migration\code\migration\config\package-names-oracle.properties
2020-01-22 09:35:12,849 INFO DBMigrationService:76 Number of Available Processors: 4
2020-01-22 09:35:12,850 INFO DBMigrationService:78 Configured simultaneous processes in the Tool : 3
2020-01-22 09:35:13,032 INFO MigrationProcessor:94 File name: D:\Migration_Output\Source\Input.sql is started
2020-01-22 09:35:13,270 INFO FileHandler:606 guessencoding command output = Error: Unable to access jarfile D:\Migration\Gauss_Tools_18_Migration\code\migration\RE_migration\target\dsctool.jar , for file= D:\Migration_Output\Source\Input.sql
2020-01-22 09:35:13,272 INFO FileHandler:625 couldn't get the encoding format, so using the default charset for D:\Migration_Output\Source\Input.sql
2020-01-22 09:35:13,272 INFO FileHandler:310 File D:\Migration_Output\Source\Input.sql will be read with charset : UTF-8
2020-01-22 09:35:13,390 INFO FileHandler:668 D:\Migration_Output\target\output\Input.sql - File already exists/Failed to create target file
2020-01-22 09:35:13,562 INFO FileHandler:606 guessencoding command output = Error: Unable to access jarfile D:\Migration\Gauss_Tools_18_Migration\code\migration\RE_migration\target\dsctool.jar , for file= D:\Migration_Output\Source\Input.sql
2020-01-22 09:35:13,563 INFO FileHandler:625 couldn't get the encoding format, so using the default charset for D:\Migration_Output\Source\Input.sql
2020-01-22 09:35:13,563 INFO FileHandler:675 File D:\Migration_Output\Source\Input.sql will be written
```

```
with charset : UTF-8
2020-01-22 09:35:13,604 INFO MigrationProcessor:139 File name: D:\Migration_Output\Source\Input.sql is processed successfully
2020-01-22 09:35:13,605 INFO MigrationService:147 Total number of files in Input folder : 3
2020-01-22 09:35:13,605 INFO MigrationService:148 Total number of queries : 1
22020-01-22 09:35:13,607 INFO PropertyLoader:164 Successfully updated Property file : D:\Migration\Gauss_Tools_18_Migration\code\migration\config\global-temp-tables.properties
2020-01-22 09:35:13,630 INFO PropertyLoader:164 Successfully updated Property file : D:\Migration\Gauss_Tools_18_Migration\code\migration\config\create-types-UDT.properties
2020-01-22 09:35:13,631 INFO PropertyLoader:164 Successfully updated Property file : D:\Migration\Gauss_Tools_18_Migration\code\migration\config\package-names-oracle.properties
2020-01-22 09:35:13,632 INFO CLMigrationUtility:305 Log file : dsc.log and the file is present in the path : D:\Migration_Output\log
2020-01-22 09:35:13,632 INFO CLMigrationUtility:312 DSC process end time : Wed Jan 22 09:35:13 GMT +05:30 2020
2020-01-22 09:35:13,632 INFO CLMigrationUtility:217 Total process time : 2842 seconds
```

错误日志

DSC仅将迁移过程中发生的错误记录到DSCError.log文件中。该文件位于log文件夹中。DSCError.log文件包含这些错误的日期、时间，文件详细信息（如文件名），以及查询位置等信息。错误日志的记录级别为ERROR。

DSCError.log的文件结构如下：

```
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
```

成功读

在DSC读取文件之后，该文件将被记录日志以进行跟踪。在某些情况下，用户可通过这些日志获取文件执行状态的信息。该文件位于log文件夹中。日志文件包括日期、时间、文件名等详细信息。此日志文件的日志记录级别为INFO。

successRead.log的文件结构如下：

```
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.
2017-07-21 14:13:02,034 INFO readlogger:213 /home/testmigration/Documentation/sampleRownum.sql is read successfully.
2017-07-21 14:13:02,578 INFO readlogger:213 /home/testmigration/Documentation/samp.sql is read successfully.
2017-07-21 14:13:03,145 INFO readlogger:213 /home/testmigration/Documentation/2.6BuildInputs/testWithNodataSamples.sql is read successfully.
```

成功写

DSC读取、处理文件并将输出写入磁盘。这个过程被记录到成功写日志文件中。在某些情况下，用户可通过此文件了解哪些文件已处理成功。在重新运行的情况下，用户

可以跳过这些文件运行剩余的文件。该文件位于log文件夹中。日志文件包括日期、时间、文件名等详细信息。此日志文件的日志记录级别为INFO。

successWrite.log的文件结构如下：

```
2017-07-21 14:13:00,616 INFO writelogger:595 /home/testmigration/Documentation/is not in.sql has written successfully.
2017-07-21 14:13:01,055 INFO writelogger:595 /home/testmigration/Documentation/date quotes.sql has written successfully.
2017-07-21 14:13:01,569 INFO writelogger:595 /home/testmigration/Documentation/column alias replace.sql has written successfully.
2017-07-21 14:13:02,055 INFO writelogger:595 /home/testmigration/Documentation/sampleRownum.sql has written successfully.
2017-07-21 14:13:02,597 INFO writelogger:595 /home/testmigration/Documentation/samp.sql has written successfully.
2017-07-21 14:13:03,178 INFO writelogger:595 /home/testmigration/Documentation/
testWithNodataSamples.sql has written successfully.
```

5.6.3 Perl 迁移日志

Perl迁移时，DSC将所有日志信息写入perlDSC.log文件。

说明

DSC通过调用SQL来迁移Perl文件中的SQL脚本，因此支持以下**SQL迁移日志**：

- 活动日志
- 错误日志
- 成功读
- 成功写

日志级别

可以使用logging-level参数来配置perl迁移日志的记录级别。

日志记录

DSC将所有日志、告警和错误信息保存到log文件夹下的perlDSC.log文件中。日志文件包含执行迁移的用户、迁移的文件、时间戳等详细信息。

perlDSC.log的文件结构如下：

```
2018-07-08 13:35:10 INFO teradatacore.pm:1316 Extracting SQL contents from perl files started
2018-07-08 13:35:10 INFO teradatacore.pm:1329 Extracting SQL contents from perl files completed
2018-07-08 13:35:10 INFO teradatacore.pm:1331 Migrating SQL files
2018-07-08 13:35:12 INFO teradatacore.pm:1348 Migrating SQL files completed
2018-07-08 13:35:12 INFO teradatacore.pm:1349 Merging migrated SQL contents to perl files started
2018-07-08 13:35:12 INFO teradatacore.pm:1362 Merging migrated SQL contents to perl files completed
2018-07-08 13:35:12 INFO teradatacore.pm:1364 Perl file migration completed
2018-07-08 13:35:32 INFO teradatacore.pm:1316 Extracting SQL contents from perl files started
2018-07-08 13:35:58 ERROR teradatacore.pm:426 opendir ../../../../perltest/ failed
2018-07-08 13:36:17 INFO teradatacore.pm:1316 Extracting SQL contents from perl files started
2018-07-08 13:38:21 INFO teradatacore.pm:1329 Extracting SQL contents from perl files completed
2018-07-08 13:38:21 INFO teradatacore.pm:1331 Migrating SQL files
2018-07-08 13:38:22 INFO teradatacore.pm:1348 Migrating SQL files completed
2018-07-08 13:38:22 INFO teradatacore.pm:1349 Merging migrated SQL contents to perl files started
2018-07-08 13:38:37 ERROR teradatacore.pm:1044 Directory ../../../../perltest/ should have 700, but has  0 permission
2018-07-08 13:38:53 ERROR teradatacore.pm:1241 Another migration process is running on same folder, re-execute after the process has completed
2018-07-08 13:39:01 INFO teradatacore.pm:1316 Extracting SQL contents from perl files started
2018-07-08 13:39:51 INFO teradatacore.pm:1329 Extracting SQL contents from perl files completed
2018-07-08 13:39:51 INFO teradatacore.pm:1331 Migrating SQL files
```

```
2018-07-08 13:39:53 INFO teradatacore.pm:1348 Migrating SQL files completed
2018-07-08 13:39:54 INFO teradatacore.pm:1349 Merging migrated SQL contents to perl files started
2018-07-08 13:39:55 INFO teradatacore.pm:1362 Merging migrated SQL contents to perl files completed
2018-07-08 13:39:57 INFO teradatacore.pm:1364 Perl file migration completed
```

5.7 DSC 常见问题

本章介绍常见问题。

问题1：在安装过程中，提示“ Root privileged users are not allowed to install the DSC for Linux. ”应如何处理?

答：拥有root权限的用户不得在Linux中安装和执行DSC。建议使用没有root权限的用户来安装和操作DSC。

问题2：如何配置DSC，以便Teradata支持GaussDB 200 V100R002C60版本?

答：执行以下步骤设置表变量值，以支持当前GaussDB 200 V100R002C60版本：

1. 打开TOOL_HOME路径下config文件夹中的Teradata features-teradata.properties文件。
2. 根据需要修改下列变量：
   - VOLATILE
   - PRIMARY INDEX

     例如：

     ```
     VOLATILE=UNLOGGED | LOCAL TEMPORARY
     PRIMARY INDEX=ONE | MANY
     ```

说明

VOLATILE变量的默认值为LOCAL TEMPORARY，PRIMARY INDEX变量的默认值为MANY。

5.8 故障处理

本章介绍使用DSC时可能遇到的问题，并提供故障处理步骤。

下表列举了常见故障的问题现象、原因、解决方案。

表 5-19 错误消息参考

问题现象	原因及解决方案
Error occurred while formatting! Returning unformatted SQL: SELECT count(*) FROM table_temp;	**原因：**可能原因为输出文件中左右括号数量不一致。 **解决方案：**确保文件中所有左右括号匹配。
ERROR QueryConversionUtility:249 Query is not converted as it contains unsupported keyword: LAST	**原因：**输入的查询文件包含一个不支持的关键词。 **解决方案：**确保要迁移的脚本中不含有不支持的关键词。

问题现象	原因及解决方案
Disk is almost full. Please clear the space and re-run the tool.	**原因：**磁盘空间不足。 **解决方案：**从磁盘中释放空间然后重试。
Please enter valid input parameters, Kindly refer the user manual to execute.	**原因：**可能原因为： 1. 未输入有效参数。 2. 缩写关键字为小写。 **解决方案：** 1. 迁移时提供全部必选参数。 2. 确保所有缩写关键字为大写。
No SQL files found in input folder. Hence stopping migration.	**原因：**迁移过程中输入的文件夹中不存在有效SQL文件。 **解决方案：**确保要迁移的SQL文件存在输入的文件夹中。 有关详情，请参见**执行DSC进行语法迁移**。
Migration Application failed to start : Currently we are not supporting this Database : *<database-name>*	**原因：**源数据库参数中提到的数据库名称不正确。 **解决方案：**DSC仅支持Teradata或Oracle作为源数据库参数的值。
Output folder is not set. Please enter an output folder and refer the user manual for syntax.	**原因：**未指定输出文件夹路径。 **解决方案：**指定输出文件夹参数的有效路径。
java.lang.OutOfMemoryError: GC overhead limit exceeded.	**原因：**在迁移操作期间，如果内存使用超过设置的值，DSC将有该错误提示并退出。 **解决方案：**可通过更改application.properties配置文件中的initialJVMMemory和maxJVMMemory的值，以分配更多内存。

问题现象	原因及解决方案
ascii “****” does not map to charset	**原因：**DSC无法检测输入文件的编码格式，且系统区域设置的字符集与输入文件的字符集不匹配。于是，系统上报告警。 **解决方案：**将encodingFormat参数设为实际编码值，并再次执行。 示例： testmigration@BLR1000026522:~/18.1_RETEST/DSC/scripts/teradata> perl sqlTDtoGS.pl -i ../../PERL -o ../../PERL_OUT/ -m /home/testmigration/18.1_FORMAT_RETEST/sep6thpackage/DSC 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 ********** Schema Conversion Started ********** DSC process start time : Mon Jan 20 17:24:49 IST 2020 Statement count progress 100% completed [FILE(1/1)] Schema Conversion Progress 100% completed ********** Total number of files in input folder : 1 ********** Log file path :...../DSC/DSC/log/dsc.log DSC process end time : Mon Jan 20 17:24:49 IST 2020 DSC total process time : 0 seconds ********** Schema Conversion Completed **********

错误码

表 5-20 错误码

错误码	错误信息
Teradata	
DSC_ERR_003_001	Query/statement is not supported since the Teradata view "dbc.indices" is supported only for the indextype P and Q.

错误码	错误信息
DSC_ERR_003_002	Error in Bteq processing. Something went wrong while processing the BTEQ commands.
DSC_ERR_003_003	Query/statement is not supported in ddl DSC. Please check the same and refer user manual for the supported feature list.
DSC_ERR_003_004	Unsupported format decimal format like ZZZ99Z, ZZZ.ZZ9.
DSC_ERR_003_005	The tool does not support the "IN/NOT IN to EXISTS/NOT EXISTS conversion" for the query in which its outer query refers multiple tables and the column(s) specified with IN / NOT IN operator do not have table reference.
DSC_ERR_003_006	The tool does not support the query in which its outer query refers multiple tables and the column(s) specified with IN / NOT IN operator do not have table reference.
DSC_ERR_003_007	Primary Index without column is not supported.
DSC_ERR_003_008	TeradataQuerySplitter config file contains list is not supported.
DSC_ERR_003_009	Gauss does not support WITH CHECK OPTION in CREATE VIEW. Please enable the config_param tdMigrateVIEWCHECKOPTION to comment the WITH CHECK OPTION syntax in the statement.
DSC_ERR_003_010	Gauss does not have an equivalent syntax for CHARACTER SET & CASE SPECIFIC option in column-level. Please enable the config_param tdMigrateCharsetCase to comment the CHARACTER SET & CASE SPECIFIC option syntax in the statement.
DSC_ERR_003_011	Gauss does not have equivalent syntax for LOCK option in CREATE VIEW and INSERT statement. You can rewrite this statement or set the configuration parameter tdMigrateLOCKOption to TRUE to comment the LOCK syntax in this statement.

错误码	错误信息
DSC_ERR_003_012	Invalid width (Number of rows) parameter in MDIFF function.
DSC_ERR_003_013	First 2 parameters should be present in MDIFF function.
DSC_ERR_003_014	Query/statement is not supported as ORDER BY clause is not present in TOP WITH TIES. Please check the same and refer user manual.
DSC_ERR_003_015	Column mismatch for the TITLE conversion.
DSC_ERR_003_016	Query/statement is not supported as same Table alias is addressed in both inner and outer query. Please check the same and refer user manual for the supported feature list.
DSC_ERR_003_017	Sub query list does not have columns.
DSC_ERR_003_018	Number of expressions specified in the outer query does not match with inner query.
DSC_ERR_003_019	Error while loading the .RUN FILE from given location.
DSC_ERR_003_020	Unable to delete the file, file not found.
DSC_ERR_003_021	Unable to delete the file, failed with IOEXception.
DSC_ERR_003_022	Please specify the value for environment_file_path parameter in features-teradata.properties.
Application	
DSC_ERR_004_001	Application has timed out, exceeded the hours specified in the config file. Please configure the Timeout parameter in the application.properties to higher value.
DSC_ERR_004_002	Error while loading the property files from config directory.
DSC_ERR_004_003	Error while loading the property files from config directory, directory is not readable.
DSC_ERR_004_004	Error while loading the property file.

错误码	错误信息
DSC_ERR_004_005	Unable to load the JSON file.
DSC_ERR_004_006	DSC tool does not support this Conversion type provided.
DSC_ERR_004_007	Error occurred while framing output replacement query.
DSC_ERR_004_008	Invalid index value while parsing the script.
DSC_ERR_004_009	Error in conversion process, unable to convert the script.
DSC_ERR_004_010	No SQL files found in the input directory with the extension specified in the fileExtension property in application.properties.
DSC_ERR_004_011	The query length parameter (MaxSqlLen) value is not valid.
DSC_ERR_004_012	Since the input folder has write privileges to Group and/or Others, process is stopped due to security reason.
DSC_ERR_004_013	Since the output directory has write privileges to Group and/or Others, process is stopped due to security reason.
DSC_ERR_004_014	Disk is almost full. Please clear the space and re-run the tool.
DSC_ERR_004_015	DSC has been cancelled as configured by the user.
DSC_ERR_004_016	Error occurred while formatting the sql scripts.
DSC_ERR_004_017	Invalid index specified for fetching the element from list while formatting the scripts
DSC_ERR_004_018	Error occurred while converting from string to integer.
DSC_ERR_004_019	Input File is modified while DSC is in progress.
DSC_ERR_004_020	Process is null, unable to read encoding format.

错误码	错误信息
DSC_ERR_004_021	Target File does not have write permissions.
DSC_ERR_004_022	The target directory does not have write privileges to Group and/or Others, process is stopped due to security reason.
DSC_ERR_004_023	PL/SQL object contains incorrect DDL/ Query. Please check the script for the query position specified in the log.
DSC_ERR_004_024	PreQueryValidation failed due to bracket mismatch or invalid terminator.
DSC_ERR_004_025	Conversion task name is not valid.
DSC_ERR_004_026	Database entered by the user is not supported by the DSC tool.
DSC_ERR_004_027	Gauss db password should not be empty.
DSC_ERR_004_028	Gauss db password should not be empty.
DSC_ERR_004_029	Target db entered in the Gaussdb.properties is not valid.
DSC_ERR_004_030	User name entered in the Gaussdb.properties is empty.
DSC_ERR_004_031	Port entered in the Gaussdb.properties is not valid.
DSC_ERR_004_032	IP entered in the Gaussdb.properties is not valid.
DSC_ERR_004_033	Database name entered in the Gaussdb.properties is empty.
DSC_ERR_004_034	DSC Application failed to start.
DSC_ERR_004_035	Since the environment variable path has write privileges to Group and/or Others, process is stopped due to security reason.
DSC_ERR_004_036	Error while loading environment parameter File.

错误码	错误信息
DSC_ERR_004_037	Invalid input (empty/space/string value) for the parameter NoOfThreads in application.properties. Hence taking the default processes.
DSC_ERR_004_038	Input for the parameter NoOfThreads in application.properties is less than 1. Hence taking the default processes.
DSC_ERR_004_039	Error in processing the DDL query.
DSC_ERR_004_040	Error in processing the PL/SQL query.
DSC_ERR_004_041	Error in post processing the query.
DSC_ERR_004_042	Invalid application timeout value, default to 4 hours.
DSC_ERR_004_043	Error in writing the output file.
DSC_ERR_004_044	Error in reading the input file.
DSC_ERR_004_045	No valid files found in the input directory for migration.
DSC_ERR_004_046	Query is not converted as it contains unsupported keyword.
DSC_ERR_004_047	Error while reading the property.
DSC_ERR_004_048	PreQueryValidation failed due to query exceeds maximum length (MaxSqlLen config parameter).
DSC_ERR_004_049	Thread count entered in the Gaussdb.properties is not valid.
Wrapper	
DSC_ERR_005_003	Reading file Failed with error: File not found Exception.
DSC_ERR_005_004	Reading file Failed with error: IOException.
DSC_ERR_005_005	Root privileged users are not allowed to execute the DSC tool.
DSC_ERR_005_006	Error while getting the id of os user used to execute the DSC tool.
DSC_ERR_005_007	Arguments specified is not valid, please check the user manual for the command line arguments.

错误码	错误信息
DSC_ERR_005_008	File name is not specified for reading the encoding type.
DSC_ERR_005_009	Invalid argument specified for the encoding parameter.
DSC_ERR_005_010	Source database is not set. Please enter a valid source db and refer the user manual for syntax.
DSC_ERR_005_011	Commandline database specified for source to target is not supported by the DSC tool.
DSC_ERR_005_012	Error in loading config file with IOException.
DSC_ERR_005_013	Initial JVM memory is greater than maximum JVM memory.
DSC_ERR_005_014	Invalid value specified for configValue.
DSC_ERR_005_015	Invalid source database specified for source-db option.
DSC_ERR_005_016	Invalid target database specified for target-db option.
DSC_ERR_005_017	Invalid conversion type specified for dsc-type option.
DSC_ERR_005_018	Invalid application language specified for application-lang option.
DSC_ERR_005_019	Conversion-type should be DDL for application-lang type as perl.
DSC_ERR_005_020	Source-db should be teradata for application-lang type as perl.
DSC_ERR_005_021	Please use "-VN [V1R7 | V1R8_330]" or "--version-number [V1R7 | V1R8_330]" to specify the kernel version which can be either V1R7 or V1R8_330.
DSC_ERR_005_022	Input directory does not exist.
DSC_ERR_005_023	Getting path for input directory failed with IOException.
DSC_ERR_005_024	Getting path for output directory failed with IOException.
DSC_ERR_005_025	Setting file permission for output directory failed with IOException.

错误码	错误信息
DSC_ERR_005_026	Creating output directory failed.
DSC_ERR_005_027	Setting file permissions for log directory/file failed with FileException.
DSC_ERR_005_028	Error while connecting to GaussDB, Failed with error.
DSC_ERR_005_029	Error occurred due to file permission while creating or executing the file.
DSC_ERR_005_030	No arguments specified in the commandline.
DSC_ERR_005_031	Error occurred in creating output directory.

5.9 DSC 术语表

下表包含缩略语、术语及其说明。

术语	描述
C	
公用表表达式(CTE)	公用表表达式是一个在查询中定义的临时命名结果集，仅可用于更大的查询范围。
D	
数据库(DB)	数据库是一组相关信息的集合，通常是为了使通用的检索变得简单和高效而组织起来的。 数据库属性： ● 数据库名称。 ● Endian文件格式（BIG_ENDIAN大端或LITTLE_ENDIAN小端）。 ● 关系。 ● 不存在无关系的数据库。
数据库管理员(DBA)	数据库管理员是负责组织中数据库的安装、配置、升级、管理、监控和维护的人员。 该角色包括开发和设计数据库策略、监控和优化数据库性能和容量，以及规划未来的扩展需求。数据库管理员亦可计划、协调和实施安全措施，以保障资料库的安全。

术语	描述
E	
编码	在信息处理中，编码是一种规则系统，它把字母、单词、声音、图像或手势等信息转换成另一种规则。有时，它以缩短或秘密的形式通过通道进行通讯或存储在介质中。
I	
索引	数据库管理系统中的一种有序数据结构，可加速表内数据的查询和更新。
M	
迁移	迁移是指将源数据库（如Teradata）中的脚本、查询、模式、数据等迁移到目标数据库（如GaussDB (DWS)）。
元数据	元数据是关于数据的数据。元数据定义了数据的属性，用于指定数据的存储位置、历史数据、检索资源数据、记录信息等。
O	
操作系统(OS)	操作系统是管理计算机中所有其他程序的程序，这些程序最初通过引导程序加载到计算机中。
Q	
查询	查询是向数据库发出的信息请求。查询执行SQL语句，并返回该语句定义的结果集。
S	
结构化查询语言(SQL)	一种编程语言，广泛用于访问、更新、管理和查询关系数据库中的数据。
模式	模式是数据库管理系统支持的正式语言中描述的结构。它是指数据的组织，描述数据库是如何构建的。（在关系数据库中，它描述了如何将数据库划分为表。）
T	
Teradata	Teradata是一种关系数据库管理系统。它可用于同时运行多个复杂查询。它支持使用SQL的即席查询，并广泛用于管理大型仓储操作。

术语	描述
表	紧密相关的列的集合。表由包含相同列的不同值的行组成。
V	
视图	视图限制对表的特定行或列的访问。视图可以从一个或多个表中创建，并且由用于创建视图的查询决定。

6 DataCheck

6.1 DataCheck 简介

6.1.1 概述

当用户选择切换到DWS数据库后可能会面临数据库的迁移任务，数据库迁移包括用户数据迁移、sql脚本迁移以及迁移之后数据校对工作，其中，数据校对是保障数据库迁移完备的重要环节。

DataCheck是一款运行在Linux或Windows操作系统上的命令行工具，致力于向用户提供简单、快速、可靠的数据校对服务，通过连接源端数据库以及目标dws数据库，将源端数据库和目标端DWS数据库中的表数据进行校对，保证用户数据迁移前后的一致性。

DataCheck需要连接数据库，可在离线模式下实现数据校对，校对结果会依次写入Excel表格中，并用日志记录操作过程中发生的错误，便于快速定位问题。

数据校验支持的源端数据库

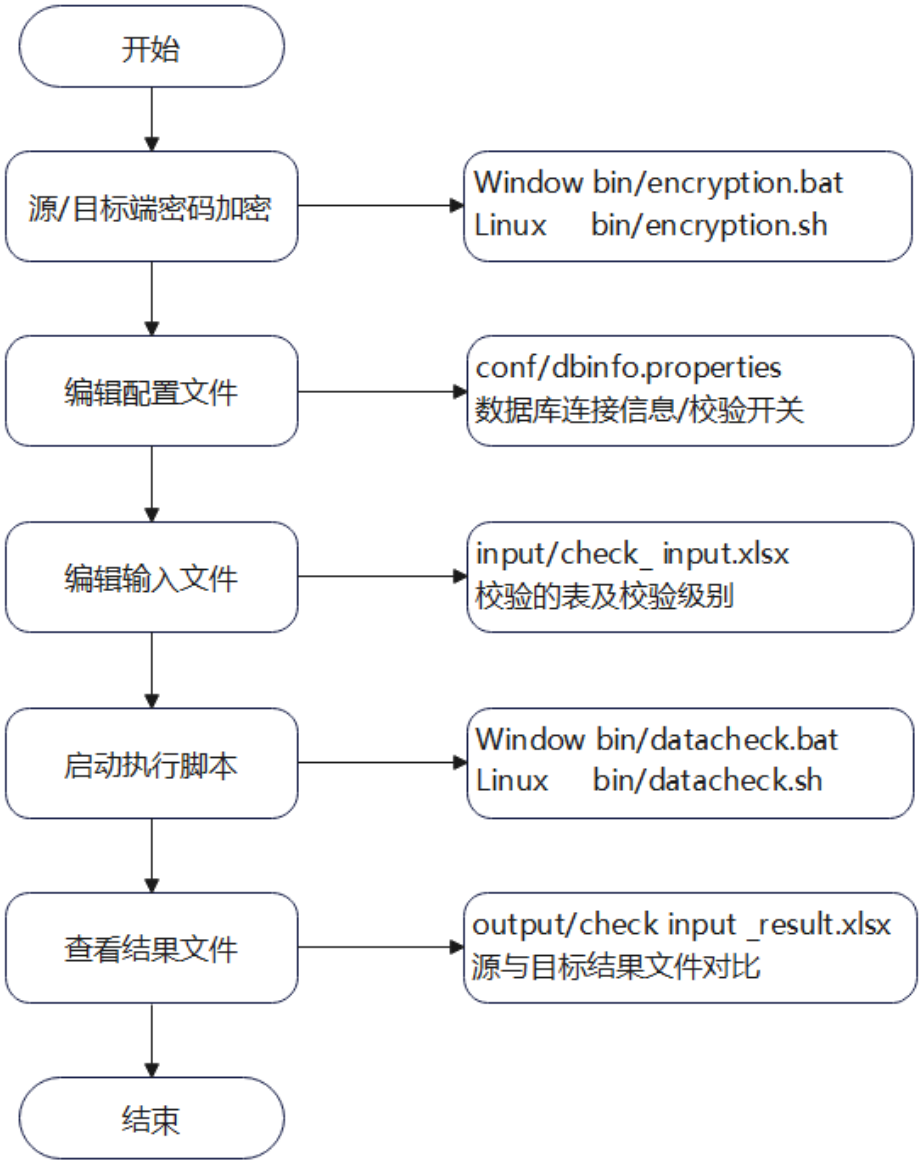
- MySQL(包括 AnalyticDB for MySQL)
- PostgreSQL
- DWS

数据校验流程

DataCheck流程如下：

1. 下载DataCheck的工具包到Linux或Windows服务器并解压。
2. 执行加密命令对源端/目标端数据库登录密码进行加密。
3. 配置dbinfo.properties文件，包含源数据库和目标数据库的相关连接信息以及函数开关信息。
4. 编辑check_input.xlsx文件，输入schema、源数据库表名和dws表名以及校验级别等参数。
5. 执行DataCheck启动命令进行数据校验，校验结果保存在check_input_result.xlsx中。

图 6-1 DataCheck 流程图



6.1.2 运行环境

支持的数据库

DataCheck支持的源数据库如所示。

表 6-1 支持的源数据库

数据库名称	数据库版本
MySQL	8.0
PostgreSQL	42.6.0
DWS	8.1.0及以上集群版本

DataCheck支持的目标数据库如所示。

表 6-2 支持的目标数据库

数据库名称	数据库版本
DWS	8.1.0及以上集群版本

硬件要求

DataCheck对硬件的要求如表6-3所示。

表 6-3 DataCheck 硬件环境要求

硬件	配置
CPU	AMD或Intel Pentium （最小频率：500 MHz）
最小内存	1 GB
磁盘空间	1 GB

软件要求

操作系统要求

DataCheck兼容的操作系统如表6-4所示。

表 6-4 兼容的操作系统

服务器	操作系统	版本
通用x86服务器	SUSE Linux Enterprise Server 11及以上	SUSE Linux Enterprise Server 11 SP1及以上
	RHEL	RedHat6.4及以上
	CentOS	CentOS6.4及以上
	Windows	7.0，10，11

其他软件要求

DataCheck对其他软件版本的要求如表6-5所示。

表 6-5 其他软件要求

软件	用途
JDK 1.8或JRE1.8	运行DataCheck工具。

6.2 DataCheck 基本功能

统计值校验

- 支持源端为DWS，MySQL，PostgreSQL，BigQuery等数据库与目标端为DWS数据库的数据校验。
- 支持通用类型字段校验：数值、时间、字符类型。
- 支持校验级别设置：包括high、middle、low三种。
- 支持指定schema、表名、列名进行校验。
- 支持指定记录的校验范围，默认为校验所有记录。
- 校验方式涉及COUNT(*)、MAX、MIN、SUM以及抽样明细校验等方式。
- 输出校验结果和相关校验明细说明。

表 6-6 数据校验级别说明

校验级别	校验说明	校验相关语法
低	数据数量校验	条数校验: COUNT(*)
中	• 数据数量校验 • 数值类型校验	• 条数校验: COUNT(*) • 数值校验: MAX, MIN, SUM
高	• 数据数量校验 • 数值类型校验 • 日期类型校验 • 字符类型校验	• 条数校验: COUNT(*) • 数值校验: MAX, MIN, SUM • 日期校验: MAX, MIN • 字符校验：order by limit 1000，读出数据并校验内容是否相同。

元数据校验

- 支持源端为DWS，MySQL，PostgreSQL，BigQuery等数据库与目标端为DWS数据库的表定义校验。
- 支持四大校验类型：字符、整数、小数、时间（包含日期）。
- 列名相同且类型一致，则校验通过。

数据精确对比

- 支持源端为DWS，MySQL，PostgreSQL，BigQuery等数据库与目标端为DWS数据库的数据精确对比。

- 根据主键或指定的列标识的唯一记录，从两端表中查询数据逐列比对，输出比对结果，包括源端或DWS端多出的记录和记录不同的列值。

注意

- 数据精确比对，比较耗执行机资源，同时也占用两端数据库的负载，应在业务空闲时执行。
- 精确比对需要指定分批查询数据的条数（默认1000条）和出现差异的结果的数量（默认100条，达到此阈值时精确比对中止）。

6.3 下载并安装 DataCheck

前提条件

- 服务器：Linux或Windows服务器，支持64位操作系统。
- JRE或JDK：系统已安装JDK 1.8或JRE 1.8。
- 网络环境：安装、运行DataCheck工具的服务器，需要与待连接的数据库的网络是互通的。

下载 DataCheck 工具

下载DataCheck客户端软件，请联系技术支持工程师。

安装 DataCheck 工具

DataCheck是一款运行在Linux或Windows操作系统上的命令行工具，可免安装使用，下载软件包后，用户解压软件包即可使用。

Windows:

步骤1 解压DataCheck-*.zip包。

得到DataCheck-*文件夹。

说明

解压DataCheck-*时，可根据需要选择任意文件夹进行解压。

步骤2 进入DataCheck-*目录。

步骤3 找到并查看DataCheck目录中的文件。

解压出来的文件夹和文件说明如表6-7所示。

----结束

Linux操作系统:

步骤1 从DataCheck-*.zip包中提取文件。

```
unzip DataCheck-*.zip
```

步骤2 进入DataCheck目录。

```
cd DataCheck-*
```

步骤3 查看DataCheck目录中的文件。

```
ls
conf   lib   bin   check_input.xlsx
```

----结束

表 6-7 DataCheck 目录

文件或文件夹		说明
DataCheck	bin	存放校验工具启动脚本。 Windows版本：datacheck.bat Linux版本：datacheck.sh
	conf	配置文件，进行源数据库和目的数据库的连接配置和日志打印设置。
	lib	保存校验工具运行所需的相关jar包。
	check_input.xlsx	1、待校验的表信息，包括Schema名、表名、列名等 2、记录用户的校验级别信息和校验规则。已支持3种级别校验，包括high、middle、low，默认为low。
	logs	压缩包中不包含该文件，校验工具执行后自动生成，记录工具运行过程日志。
	check_input_result.xlsx	压缩包中不包含该文件，校验工具执行后会在check_input.xlsx相同路径下生成校验结果文件。

6.4 配置 DataCheck

6.4.1 dbinfo.properties 配置

dbinfo.properties文件中包括一系列应用配置参数，用于连接源端数据库和目标dws数据库，该文件中的参数为通用参数。

设置方法如下。

步骤1 打开conf文件夹中的dbinfo.properties文件。

步骤2 根据实际需要修改dbinfo.properties文件中参数的值。

dbinfo.properties文件中的参数说明见表6-8。

说明

- 参数值不区分大小写。
- 除了列出的参数外，不得更改其他参数值。

步骤3 保存后退出。

----结束

表 6-8 dbinfo.properties 文件的配置参数

参数	说明	取值范围	默认值	样例
src.dbtype	源端数据库类型。	• mysql • pg • oracle • dws • hive • mrs_hive • hologres • greenplum • redshift • bigquery • elasticsearch • synapse	MySQL	src.dbtype=mysql
src.dbname	源端数据库名称。	NA	sys	src.dbname=sys
src.ip	源端数据库ip地址。	NA	NA	src.ip=100.xx.xx.47
src.port	源端数据库端口。	NA	3306	src.port=3306
src.username	源端数据库用户名。	NA	root	src.username=root
src.passwd	源端数据库密码。	NA	NA	src.passwd=123456
dws.dbtype	目标端dws数据库类型。	dws	dws	dws.dbtype=dws
dws.dbname	目标端dws数据库名称。	NA	gaussdb	dws.dbname=gaussdb
dws.ip	目标端dws数据库ip地址。	NA	NA	dws.ip=100.xx.xx.186
dws.port	目标端dws数据库端口。	NA	8000	dws.port=8000
dws.username	目标端dws数据库用户名。	NA	dbadmin	dws.username=dbadmin

参数	说明	取值范围	默认值	样例
dws.passwd	目标端dws数据库密码。	NA	NA	dws.passwd=123456
config.sum.switch	数值校验：求和函数开关。	• on • off	on	config.sum.switch=on
config.data.min.switch	数值校验：最小值函数开关。	• on • off	on	config.data.min.switch=on
config.data.max.switch	数值校验：最大值函数开关。	• on • off	on	config.data.max.switch=on
config.date.min.switch	日期校验：最小值函数开关。	• on • off	on	config.date.min.switch=on
config.date.max.switch	日期校验：最大值函数开关。	• on • off	on	config.date.max.switch=on
config.collate.switch	Collate规则计算的开关配置。 on：启动；off：关闭。	• on • off	on	config.collate.switch=on
config.collate.value	Collate规则的值：“C”、“zh_CN”、“en_US”。	• C • zh_CN • en_US	C	config.dws.collate=C
src.projectid	源端的项目ID，BigQuery源需要配置。	NA	NA	src.projectid=xxxxxxx
src.oauthtype	源端的oauth类型，BigQuery源需要配置。	• 0	0	src.oauthtype
src.oauthserviceacctemail	源端的oauth邮箱，BigQuery源需要配置。	NA	NA	src.oauthserviceacctemail=xxx@sina.com
src.oauthpvtkeypath	源端的oauth认证信息文件的路径，BigQuery源需要配置。	NA	NA	src.oauthpvtkeypath =/opt/temp/analytics-di-dev-a8dfdf.json
config.batch.size	精确比对时，每批查询的数据条数。	NA	1000	config.batch.size = 1000

参数	说明	取值范围	默认值	样例
precise.result.max	精确比对结果中出现差异的最大值，达到此阈值时会终止精确比对。	NA	100	precise.result.max = 100
difference	浮点数比较时允许的误差值。	NA	0.0001	difference = 0.0001
output.file.suffix	输出结果文件的后缀，支持csv/xlsx。	• xlsx • csv	xlsx	output.file.suffix = csv

6.4.2 check_input.xlsx 配置

check_input.xlsx文件中包括用户输入信息：schema、原表名、目标表名、指定列名（缺省为全部列校验）、校验范围、校验级别（缺省为low），排除列，可以按需进行配置。

设置方法如下。

步骤1 打开文件夹中的check_input.xlsx文件。

步骤2 根据实际需要修改check_input.xlsx文件中参数的值。

check_input.xlsx文件中的参数说明见**表6-9**。

说明

- 参数值不区分大小写。
- 除了列出的参数外，不得更改其他参数值。

步骤3 保存后退出。

----结束

表 6-9 check_input.xlsx 文件的配置参数

参数	说明	取值范围	默认值	样例
Source Database Name	指定需要校验的源表所属的database 名称，**可选项**，不填写表示使用dbinfo.properties中的src.dbname。	NA	NA	mydb
Source Schema Name	指定需要校验的源表所属的schema名称，不涉及则不填写。	NA	NA	myschema
Source Table Name	源端数据库需要校验的表名，**必填项**。	NA	NA	order_info

参数	说明	取值范围	默认值	样例
Target Database Name	目标端DWS表所属的database 名称，**可选项**，不填写表示使用dbinfo.properties中的dws.dbname。	NA	NA	dws_db
Target Schema Name	目标端DWS表所属的schema名称，**必填项**。	NA	NA	dws_sch
Target Table Name	目标端dws数据库需要校验的表名，**必填项**。	NA	NA	dws_info
Check Mode	校验模式：统计值校验，精准校验，元数据校验	• Statistics • Preciseness • Metadata	• Statistics	Statistics
Src Row Range(Where sql)	指定源数据表记录的校验范围，默认为校验所有记录。	NA	ALL	where begin_time > '2020-1-1' and begin_time < '2021-1-1'
DWS Row Range(Where sql)	指定目标（DWS）表记录的校验范围，默认为校验所有记录。	NA	ALL	where begin_time > '2020-1-1' and begin_time < '2021-1-1'
Column Range	可选，指定列校验，默认为所有支持类型字段，即：数值、时间、字符类型。	NA	NA	col1,col10,col19
Check Strategy	数据校验级别。	• low • middle • high	low	low
Sort Column	可选，精确校验时根据Sort Column字段进行排序，如果Sort Column为空，则根据主键进行排序，排序后逐行对比。	NA	NA	col1,col10,col19
Column Exclude	可选，排除指定列的校验。	NA	NA	col1,col10,col19

参数	说明	取值范围	默认值	样例
Columns Wthout Sum	可选，排除指定列的sum校验。	NA	NA	col1,col10,col19

6.5 使用 DataCheck

注意事项

- 启动DataCheck前，必须配置config文件夹中dbinfo.properties文件和check_input.xlsx文件。参数配置错误会导致DataCheck执行错误。
- 如果在同一台服务器上并发进行DataCheck（由同一个或不同DataCheck执行），不同的DataCheck任务必须使用不同的check_input.xlsx文件。
- 用户在执行完DataCheck后会生成logs文件夹，可以进入logs目录查看工具执行过程中的日志，方便定位问题。

基于 Linux 使用 DataCheck

步骤1 上传工具包到Linux服务器并解压：

```
[root@dws-testng-nodelete DataCheck]# ll
total 32
drwx------ 2 root root  4096 Nov 12 20:47 bin
drwxr-xr-x 2 root root  4096 Nov 12 20:34 conf
drwxr-xr-x 2 root root  4096 Nov 12 20:34 input
drwxr-xr-x 2 root root 20480 Nov 12 20:34 lib
```

步骤2 生成数据库登录密码密文：

进入bin目录：

```
[root@dws-testng-nodelete DataCheck]# cd bin
[root@dws-testng-nodelete bin]#
[root@dws-testng-nodelete bin]#
[root@dws-testng-nodelete bin]# ll
total 16
-rwxr-xr-x 1 root root 1254 Oct 19 14:58 datacheck.bat
-rwxr-xr-x 1 root root  410 Oct 19 14:58 datacheck.sh
-rwxr-xr-x 1 root root 1322 Oct 19 14:58 encryption.bat
-rwxr-xr-x 1 root root  385 Oct 19 14:58 encryption.sh
```

执行密文生成的脚本，密文会输出。对源端和目标端数据库登录密码分别执行此脚本生成密文。

sh encryption.sh [password]

```
[root@dws-testng-nodelete bin]# sh encryption.sh  12[redacted]
B06456948116E8E77222B6C4EE5F7CF7A122E357C[redacted]
[root@dws-testng-nodelete bin]#
```

步骤3 配置conf/dbinfo.properties文件：

进入Datacheck目录下，执行vi conf/dbinfo.properties

```
[root@dws-testng-nodelete DataCheck]# cd /opt/temp/DataCheck
[root@dws-testng-nodelete DataCheck]# vi conf/dbinfo.properties
[root@dws-testng-nodelete DataCheck]#
```

配置源端和目标端的数据库连接信息，配置文件中的密码，使用上一步生成的密文。

```
[Source Database Info]
## src.dbtype support: mysql/pg/oracle/dws_src/
src.dbtype = mysql
src.dbname = db_test
src.ip = localhost
src.port = 3306
src.username = admin
src.passwd = 745675379[redacted]

src.projectid =
src.oauthtype = 0
src.oauthserviceacctemail =
src.oauthpvtkeypath =

src.jar.path =
input.file.path =

[DWS Database Info]
dws.dbtype = dws
dws.dbname = db_dws
dws.ip = 100[redacted]
dws.port = 8000
dws.username = dbadmin[redacted]
dws.passwd = 424312BAB[redacted]
```

步骤4 编辑input/check_input.xlsx文件：

复制check_input.xlsx文件到windows服务器，使用Excel软件编辑，填写要校验的表信息，保存后，上传到Linux服务器覆盖原始的文件。

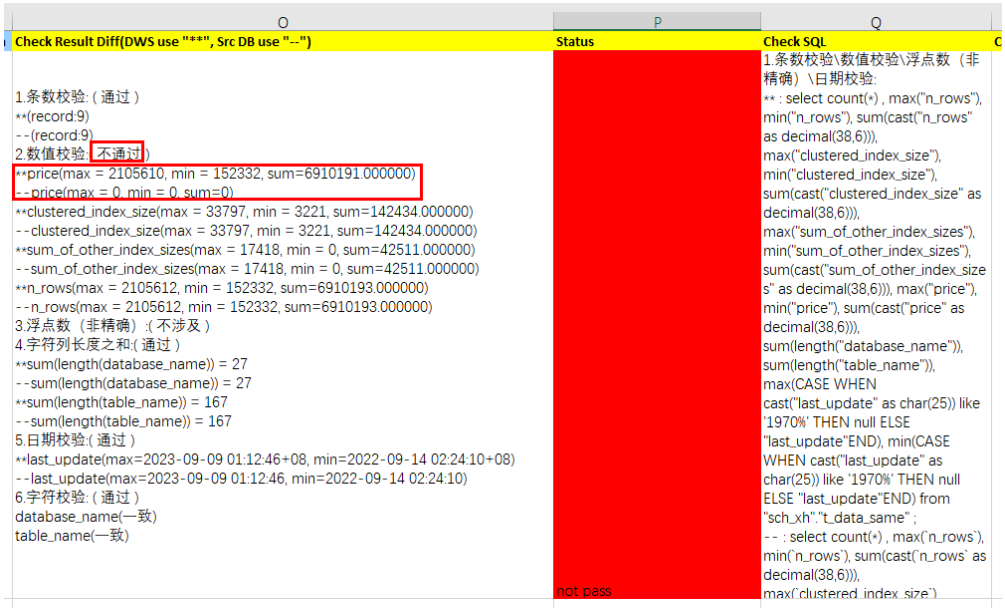
Source Database Name	Source Schema Name	Source Table Name	Target Database Name	Target Schema Name	Target Table Name	* Check Mode	Check Strategy
		t_data_20		sch_test	t_data_21	Statistics	high

步骤5 执行数据校验工具：

进入bin目录，执行启动脚本sh datacheck.sh

```
[root@dws-testng-nodelete DataCheck]# cd /opt/temp/DataCheck/bin/
[root@dws-testng-nodelete bin]# sh datacheck.sh
```

步骤6 查看校验结果 output/check_input_result.xlsx：



O	P	Q
Check Result Diff(DWS use "**", Src DB use "--")	Status	Check SQL
1.条数校验:(通过) **(record:9) --(record:9) 2.数值校验:(不通过) **price(max = 2105610, min = 152332, sum=6910191.000000) --price(max = 0, min = 0, sum=0) **clustered_index_size(max = 33797, min = 3221, sum=142434.000000) --clustered_index_size(max = 33797, min = 3221, sum=142434.000000) **sum_of_other_index_sizes(max = 17418, min = 0, sum=42511.000000) --sum_of_other_index_sizes(max = 17418, min = 0, sum=42511.000000) **n_rows(max = 2105612, min = 152332, sum=6910193.000000) --n_rows(max = 2105612, min = 152332, sum=6910193.000000) 3.浮点数（非精确）:(不涉及) 4.字符列长度之和:(通过) **sum(length(database_name)) = 27 --sum(length(database_name)) = 27 **sum(length(table_name)) = 167 --sum(length(table_name)) = 167 5.日期校验:(通过) **last_update(max=2023-09-09 01:12:46+08, min=2022-09-14 02:24:10+08) --last_update(max=2023-09-09 01:12:46, min=2022-09-14 02:24:10) 6.字符校验:(通过) database_name(一致) table_name(一致)	not pass	1.条数校验\数值校验\浮点数（非精确）\日期校验: ** : select count(*) , max("n_rows"), min("n_rows"), sum(cast("n_rows" as decimal(38,6))), max("clustered_index_size"), min("clustered_index_size"), sum(cast("clustered_index_size" as decimal(38,6))), max("sum_of_other_index_sizes"), min("sum_of_other_index_sizes"), sum(cast("sum_of_other_index_sizes" as decimal(38,6))), max("price"), min("price"), sum(cast("price" as decimal(38,6))), sum(length("database_name")), sum(length("table_name")), max(CASE WHEN cast("last_update" as char(25)) like '1970%' THEN null ELSE "last_update"END), min(CASE WHEN cast("last_update" as char(25)) like '1970%' THEN null ELSE "last_update"END) from "sch_xh"."t_data_same" ; -- : select count(*) , max(`n_rows`), min(`n_rows`), sum(cast(`n_rows` as decimal(38,6))), max(`clustered_index_size`)

步骤7 校验结果分析：

1. Status结果为No Pass代表校验未通过。
2. Check Result Diff列显示校验不通过的项，可在里面查看具体哪一列的校验不通过。
3. Check SQL中显示在数据库中执行的查询SQL。

----结束

基于 Windows 使用 DataCheck

步骤1 上传工具包到Windows服务器并解压：

bin	2024/11/13 10:31
conf	2024/11/13 9:42
input	2024/11/13 9:42
lib	2024/11/13 9:42

步骤2 生成数据库登录密码密文：

进入bin目录，启动CMD工具：

datacheck.bat	2024/10/19 14:58
datacheck.sh	2024/10/19 14:58
encryption.bat	2024/10/19 14:58
encryption.sh	2024/10/19 14:58

执行密文生成的脚本，密文会输出。对源端和目标端数据库登录密码分别执行此脚本生成密文。

encryption.bat [password]

```
D:\temp\DataCheck\bin>encryption.bat 1[redacted]
7EE79288E4BAFA94808D5540794CB1004491[redacted]

D:\temp\DataCheck\bin>encryption.bat a[redacted]
416A0EB307C290131A1C3D2AC77D36AC6D[redacted]
```

步骤3 配置conf/dbinfo.properties文件：

编辑conf目录下的dbinfo.properties文件，配置源端和目标端的数据库连接信息，配置文件中的密码，使用上一步生成的密文。

```
[Source Database Info]
## src.dbtype support: mysql/pg/oracle/dws_src
src.dbtype = mysql
src.dbname = mysql
src.ip = localhost
src.port = 3306
src.username = ***
src.passwd = *****

src.jar.path =
input.file.path =

[DWS Database Info]
## dws.dbtype support: dws
dws.dbtype = dws
dws.dbname = dbname
dws.ip = localhost
dws.port = 8000
dws.username = ***
dws.passwd = *****
```

步骤4 编辑input/check_input.xlsx文件并保存：

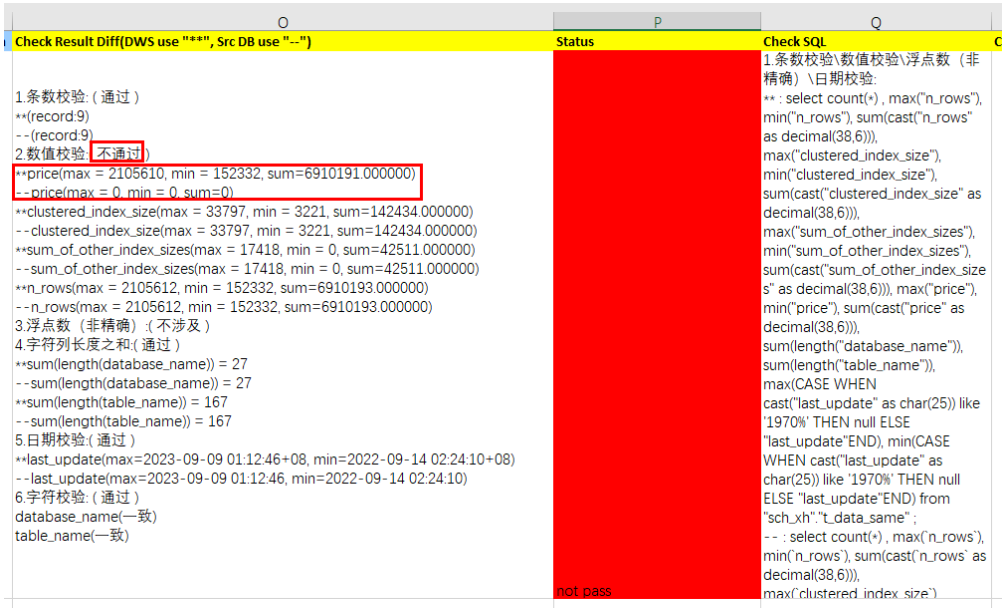
使用Excel软件编辑input/check_input.xlsx，填写要校验的表信息并保存。

Source Database Name	Source Schema Name	Source Table Name	Target Database Name	Target Schema Name	Target Table Name	* Check Mode	Check Strategy
		t_data_20		sch_test	t_data_21	Statistics	high

步骤5 执行数据校验工具 datacheck.bat：

```
D:\temp\DataCheck\bin>datacheck.bat
```

步骤6 查看校验结果 output/check_input_result.xlsx(校验结果分析同Linux场景)：



O	P	Q
Check Result Diff(DWS use "", Src DB use "--")**	**Status**	**Check SQL** C
1.条数校验:(通过) **(record:9) --(record:9) 2.数值校验:(不通过) **price(max = 2105610, min = 152332, sum=6910191.000000) --price(max = 0, min = 0, sum=0) **clustered_index_size(max = 33797, min = 3221, sum=142434.000000) --clustered_index_size(max = 33797, min = 3221, sum=142434.000000) **sum_of_other_index_sizes(max = 17418, min = 0, sum=42511.000000) --sum_of_other_index_sizes(max = 17418, min = 0, sum=42511.000000) **n_rows(max = 2105612, min = 152332, sum=6910193.000000) --n_rows(max = 2105612, min = 152332, sum=6910193.000000) 3.浮点数（非精确）:(不涉及) 4.字符列长度之和:(通过) **sum(length(database_name)) = 27 --sum(length(database_name)) = 27 **sum(length(table_name)) = 167 --sum(length(table_name)) = 167 5.日期校验:(通过) **last_update(max=2023-09-09 01:12:46+08, min=2022-09-14 02:24:10+08) --last_update(max=2023-09-09 01:12:46, min=2022-09-14 02:24:10) 6.字符校验:(通过) database_name(一致) table_name(一致)	not pass	1.条数校验\数值校验\浮点数（非精确）\日期校验: ** : select count(*) , max("n_rows"), min("n_rows"), sum(cast("n_rows" as decimal(38,6))), max("clustered_index_size"), min("clustered_index_size"), sum(cast("clustered_index_size" as decimal(38,6))), max("sum_of_other_index_sizes"), min("sum_of_other_index_sizes"), sum(cast("sum_of_other_index_sizes" as decimal(38,6))), max("price"), min("price"), sum(cast("price" as decimal(38,6))), sum(length("database_name")), sum(length("table_name")), max(CASE WHEN cast("last_update" as char(25)) like '1970%' THEN null ELSE "last_update"END), min(CASE WHEN cast("last_update" as char(25)) like '1970%' THEN null ELSE "last_update"END) from "sch_xh"."t_data_same" ; -- : select count(*) , max(`n_rows`), min(`n_rows`), sum(cast(`n_rows` as decimal(38,6))), max(`clustered_index_size`)

步骤7 校验结果分析：

1. Status结果为No Pass代表校验未通过。
2. Check Result Diff列显示校验不通过的项，可在里面查看具体哪一列的校验不通过。
3. Check SQL中显示在数据库中执行的查询SQL。

----结束

7 服务端工具

7.1 gs_dump

背景信息

gs_dump是DWS用于导出数据库相关信息的工具，用户可以自定义导出一个数据库或其中的对象（模式、表、视图等）。支持导出的数据库可以是默认数据库postgres，也可以是自定义数据库。

gs_dump工具在进行数据导出时，导出的表会被加锁，会导致阻塞读写。

gs_dump工具支持导出完整一致的数据。例如，T1时刻启动gs_dump导出A数据库，那么导出数据结果将会是T1时刻A数据库的数据状态，T1时刻之后对A数据库的修改不会被导出。

gs_dump支持将数据库信息导出至纯文本格式的SQL脚本文件或其他归档文件中。

- 纯文本格式的SQL脚本文件：包含将数据库恢复为其保存时的状态所需的SQL语句。通过gsql运行该SQL脚本文件，可以恢复数据库。即使在其他主机和其他数据库产品上，只要对SQL脚本文件稍作修改，也可以用来重建数据库。
- 归档格式文件：包含将数据库恢复为其保存时的状态所需的数据，可以是tar格式、目录归档格式或自定义归档格式，详见**表7-1**。该导出结果必须与**gs_restore**配合使用来恢复数据库，gs_restore工具在导入时，系统允许用户选择需要导入的内容，甚至可以在导入之前对等待导入的内容进行排序。

主要功能

gs_dump可以创建四种不同的导出文件格式，通过**[**-F或者--format=**]**选项指定，具体如**表7-1**所示。

表 7-1 导出文件格式

格式名称	-F的参数值	说明	建议	对应导入工具
纯文本格式	p	纯文本脚本文件包含SQL语句和命令。命令可以由gsql命令行终端程序执行，用于重新创建数据库对象并加载表数据。	小型数据库，一般推荐纯文本格式。	使用gsql工具恢复数据库对象前，可根据需要使用文本编辑器编辑纯文本导出文件。
自定义归档格式	c	一种二进制文件。支持从导出文件中恢复所有或所选数据库对象。	中型或大型数据库，推荐自定义归档格式。	使用**gs_restore**可以选择要从自定义归档导出文件中导入相应的数据库对象。
目录归档格式	d	该格式会创建一个目录，该目录包含两类文件，一类是目录文件，另一类是每个表和blob对象对应的数据文件。	-	
tar归档格式	t	tar归档文件支持从导出文件中恢复所有或所选数据库对象。tar归档格式不支持压缩且对于单独表大小应小于8GB。	-	

说明

可以使用gs_dump程序将文件压缩为纯文本或自定义归档导出文件，减少导出文件的大小。生成纯文本导出文件时，默认不压缩。生成自定义归档导出文件时，默认进行中等级别的压缩。gs_dump程序无法压缩已归档导出文件。

注意事项

禁止修改导出的文件和内容，否则可能无法恢复成功。

为了保证数据一致性和完整性，gs_dump会对需要转储的表设置共享锁。如果表在别的事务中设置了共享锁，gs_dump会等待锁释放后锁定表。如果无法在指定时间内锁定某个表，转储会失败。用户可以通过指定--lock-wait-timeout选项，自定义等待锁超时时间。

语法

```
gs_dump [OPTION]... [DBNAME]
```

说明

“dbname”前面不需要加短或长选项。“dbname”指定要连接的数据库。

例如：

不需要-d，直接指定“dbname”。

```
gs_dump -p port_number  postgres -f dump1.sql
```

或者

```
export PGDATABASE=postgres
 gs_dump -p port_number -f dump1.sql
```

环境变量： PGDATABASE

参数说明

通用参数：

- -f, --file=FILENAME

  将输出发送至指定文件或目录。如果省略该参数，则使用标准输出。如果输出格式为(-F c/-F d/-F t)时，必须指定-f参数。如果-f的参数值含有目录，要求目录对当前用户具有读写权限。

- -F, --format=c|d|t|p

  选择输出格式。格式如下：

  - p|plain：输出一个文本SQL脚本文件（默认）。
  - c|custom：输出一个自定义格式的归档，并且以目录形式输出，作为gs_restore输入信息。该格式是最灵活的输出格式，因为能手动选择，而且能在恢复过程中将归档项重新排序。该格式默认状态下会被压缩。
  - d|directory：该格式会创建一个目录，该目录包含两类文件，一类是目录文件，另一类是每个表和blob对象对应的数据文件。
  - t|tar：输出一个tar格式的归档形式，作为gs_restore输入信息。tar格式与目录格式兼容；tar格式归档形式在提取过程中会生成一个有效的目录格式归档形式。但是，tar格式不支持压缩且对于单独表有8GB的大小限制。此外，表数据项的相应排序在恢复过程中不能更改。

    输出一个tar格式的归档形式，也可以作为gsql输入信息。

- -v, --verbose

  指定verbose模式。该选项将导致gs_dump向转储文件输出详细的对象注解和启动/停止次数，向标准错误流输出处理信息。

- -V, --version

  打印gs_dump版本，然后退出。

- -Z, --compress=0-9

  指定使用的压缩比级别。

  取值范围：0~9

  - 0表示无压缩。
  - 1表示压缩比最小，处理速度最快。
  - 9表示压缩比最大，处理速度最慢。

  针对自定义归档格式，该选项指定单个表数据片段的压缩，默认方式是以中等级别进行压缩。对于文本输出，设置非零压缩级别将会导致整个输出文件被压缩（类似通过gzip进行压缩），默认不压缩。tar归档格式目前不支持压缩。

- --lock-wait-timeout=TIMEOUT

  请勿在转储刚开始时一直等待以获取共享表锁。如果无法在指定时间内锁定某个表，就选择失败。可以以任何符合SET statement_timeout的格式指定超时时间。

- -?, --help

  显示gs_dump命令行参数帮助，然后退出。

转储参数：

- -a, --data-only

  只输出数据，不输出模式(数据定义)。转储表数据、大对象和序列值。

- -b, --blobs

  该参数为扩展预留接口，不建议使用。

- -c, --clean

  在将创建数据库对象的指令输出到备份文件之前，先将清理（删除）数据库对象的指令输出到备份文件中。（如果目标数据库中没有任何对象，gs_restore工具可能会输出一些提示性的错误信息）

  该选项只对文本格式有意义。针对归档格式，可以在调用gs_restore时指定选项。

- -C, --create

  备份文件以创建数据库和连接到创建的数据库的命令开始。（如果命令脚本是这种方式执行，无所谓在运行脚本之前连接的是哪个数据库。）

  该选项只对文本格式有意义。针对归档格式，可以在调用gs_restore时指定选项。

- -E, --encoding=ENCODING

  以指定的字符集编码创建转储。默认情况下，以数据库编码创建转储。（得到相同结果的另一个办法是将环境变量“PGCLIENTENCODING”设置为所需的转储编码。）

- -n, --schema=SCHEMA

  只转储与模式名称匹配的模式，此选项包括模式本身和所有它包含的对象。如果该选项没有指定，所有在目标数据库中的非系统模式将会被转储。写入多个-n选项来选择多个模式。此外，根据gsql的\d命令所使用的相同规则，模式参数可被理解成一个pattern，所以多个模式也可以通过在该pattern中写入通配符来选择。使用通配符时，注意给pattern打引号，防止shell扩展通配符。

  说明

  - 当-n已指定时，gs_dump不会转储已选模式所附着的任何其他数据库对象。因此，无法保证某个指定模式的转储结果能够自行成功地储存到一个空数据库中。
  - 当-n指定时，非模式对象不会被转储。

  转储支持多个模式的转储。多次输入-n schemaname转储多个模式。

  例如：

  ```
  gs_dump -h host_name -p port_number postgres -f backup/bkp_shl2.sql -n sch1 -n sch2
  ```

  在上面这个例子中，sch1和sch2会被转储。

- -N, --exclude-schema=SCHEMA

  不转储任何与模式pattern匹配的模式。Pattern将参照针对-n的相同规则来理解。可以通过输入多次-N，不转储与任何pattern匹配的模式。

  当同时输入-n和-N时，会转储与至少一个-n选项匹配、与-N选项不匹配的模式。如果有-N没有-n，则不转储常规转储中与-N匹配的模式。

  转储过程支持排除多个模式。

在转储过程中，输入-N exclude schema name排除多个模式。

例如：

```
gs_dump -h host_name -p port_number postgres -f backup/bkp_shl2.sql -N sch1 -N sch2
```

在上面这个例子中，sch1和sch2在转储过程中会被排除。

- -o, --oids

  转储每个表的对象标识符（OIDs），作为表的一部分数据。该选项用于应用以某种方式（例如：外键约束方式）参照了OID列的情况。如果不是以上这种情况，请勿使用该选项。

- -O, --no-owner

  不输出设置对象的归属这样的命令，以匹配原始数据库。默认情况下，gs_dump会发出ALTER OWNER或SET SESSION AUTHORIZATION语句设置所创建的数据库对象的归属。如果脚本正在运行，该语句不会执行成功，除非是由系统管理员触发（或是拥有脚本中所有对象的同一个用户）。通过指定-O，编写一个任何用户都能存储的脚本，且该脚本会授予该用户拥有所有对象的权限。

  该选项只对文本格式有意义。针对归档格式，可以在调用gs_restore时指定选项。

- -s, --schema-only

  只转储对象定义（模式），而非数据。

- -S, --sysadmin=NAME

  该参数为扩展预留接口，不建议使用。

- -t, --table=TABLE

  指定转储的表（或视图、或序列、或外表）对象列表，可以使用多个-t选项来选择多个表，也可以使用通配符指定多个表对象。

  当使用通配符指定多个表对象时，注意给pattern打引号，防止shell扩展通配符。

  当使用-t时，-n和-N没有任何效应，这是因为由-t选择的表的转储不受那些选项的影响。

  说明

  -t参数选项个数必须小于等于100。

  如果-t参数选项个数大于100，建议使用参数--include-table-file来替换。

  当-t已指定时，gs_dump不会转储已选表所附着的任何其他数据库对象。因此，无法保证某个指定表的转储结果能够自行成功地储存到一个空数据库中。

  -t tablename只转储在默认搜索路径中可见的表。-t '*.tablename'转储数据库下所有模式下的tablename表。-t schema.table转储特定模式中的表。

  -t tablename不会导出表上的触发器信息。

  例如：

  ```
  gs_dump -h host_name -p port_number postgres -f backup/bkp_shl2.sql -t schema1.table1 -t schema2.table2
  ```

  在上面这个例子中，schema1.table1和schema2.table2会被转储。

- --include-table-file=FILENAME

  指定需要dump的表文件。

- -T, --exclude-table=TABLE

  不转储的表（或视图、或序列、或外表）对象列表，可以使用多个-t选项来选择多个表，也可以使用通配符指定多个表对象。

  当同时输入-t和-T时，会转储在-t列表中，而不在-T列表中的表对象。

  例如：

```
gs_dump -h host_name -p port_number postgres -f backup/bkp_shl2.sql -T table1 -T table2
```

在上面这个例子中，table1和table2在转储过程中会被排除。

- --exclude-table-file=FILENAME

  指定不需要dump的表文件。

  说明

  同--include-table-file，其内容格式如下：
  schema1.table1
  schema2.table2
  ......

- -x, --no-privileges|--no-acl

  防止转储访问权限（授权/撤销命令）。

- --column-inserts|--attribute-inserts

  以INSERT命令带列名（INSERT INTO表（列、…）值…）方式导出数据。这会导致恢复缓慢。但是由于该选项会针对每行生成一个独立分开的命令，所以在重新加载某行时出现的错误只会导致那行丢失，而非整个表内容。

- --disable-dollar-quoting

  该选项将禁止在函数体前使用美元符号$，并强制使用SQL标准字符串语法对其进行引用。

- --disable-triggers

  该参数为扩展预留接口，不建议使用。

- --exclude-table-data=TABLE

  指定不转储任何匹配表pattern的表这方面的数据。依照针对-t的相同规则理解该pattern。

  可多次输入--exclude-table-data来排除匹配任何pattern的表。当用户需要特定表的定义但不需要其中的数据时，这个选项很有帮助。

  排除数据库中所有表的数据，参见**--schema-only**。

- --inserts

  发出INSERT命令（而非COPY命令）时转储数据。这会导致恢复缓慢。

  但是由于该选项会针对每行生成一个独立分开的命令，所以在重新加载某行时出现的错误只会导致那行丢失，而非整个表内容。注意如果重排列顺序，可能会导致恢复整个失败。列顺序改变时，--column-inserts选项不受影响，虽然会更慢。

- --no-security-labels

  该参数为扩展预留接口，不建议使用。

- --no-tablespaces

  该参数在8.2.0.100版本中已废弃，为兼容历史版本功能保留该函数。

  不输出选择表空间的命令。使用该选项，无论默认表空间是哪个，在恢复过程中所有对象都会被创建。

  该选项只对文本格式有意义。针对归档格式，可以在调用gs_restore时指定选项。

- --no-unlogged-table-data

  该参数为扩展预留接口，不建议使用。

- --non-lock-table

  该参数为扩展预留接口，不建议使用。

- --quote-all-identifiers

  强制对所有标识符加引号。为了向后续版本迁移，且其中可能涉及引入额外关键词，在转储相应数据库时该选项会有帮助。

- --section=SECTION

  指定已转储的名称区段（pre-data、data、和post-data）。

- --serializable-deferrable

  转储过程中使用可串行化事务，以确保所使用的快照与之后的数据库状态一致；要实现该操作需要在无异常状况的事务流中等待某个点，因为这样才能保证转储成功，避免引起其他事务出现serialization_failure要重新再做。

  但是该选项对于灾难恢复没有益处。对于在原始数据库进行升级的时候，加载一个数据库的拷贝作为报告或其他只读加载共享的转储是有帮助的。没有这个选项，转储会反映一个与任何事务最终提交的序列化执行不一致的状态。

  如果当gs_dump启动时，读写事务仍处于非活动状态，即便使用该选项也不会对其产生影响。如果读写事务处于活动状态，转储的开始时间可能会延迟一段不确定的时间。

- --use-set-session-authorization

  输出符合SQL标准的SET SESSION AUTHORIZATION命令而不是ALTER OWNER命令来确定对象所有权。这样令转储更加符合标准，但是如果转储文件中的对象的历史有些问题，那么可能不能正确恢复。并且，使用SET SESSION AUTHORIZATION的转储需要数据库系统管理员的权限才能转储成功，而ALTER OWNER需要的权限则低得多。

- --with-encryption=AES128

  指定转储数据需用AES128进行加密。

- --with-key=KEY

  AES128密钥长度必须是16字节。

- --include-nodes

  将TO NODE/TO GROUP语句包含在已转储的CREATE TABLE/CREATE FOREIGN TABLE语句中。该参数只对HDFS表和外表生效。

- --include-extensions

  在转储中包含扩展。

- --include-depend-objs

  备份结果包含依赖于指定对象的对象信息。该参数需要同-t/--include-table-file参数关联使用才会生效。

- --exclude-self

  备份结果不包含指定对象自身的信息。该参数需要同-t/--include-table-file参数关联使用才会生效。

- --cstore-fine-disaster（已废弃）

  选择此参数时，如果dump一张表级参数fine_disaster_table_role为“primary”的表，会得到一个表级参数fine_disaster_table_role为“standby”的表定义。

  该参数在8.2.1版本废弃。

- --only-publications

  指定该参数时，只dump当前数据库的所有发布（publication）的定义。该参数仅8.2.1及以上集群版本支持。

- --no-comment

在转储中不包含对象注释信息。该参数仅9.1.0.100及以上集群版本支持。

- --dont-overwrite-file

  文本、tar、以及自定义格式情况下会重写现有文件。这对目录格式不适用。

  例如：

  设想这样一种情景，即当前目录下backup.sql已存在。如果在输入命令中输入-f backup.sql选项时，当前目录恰好也生成backup.sql，文件就会被重写。

  如果备份文件已存在，且输入--dont-overwrite-file选项，则会报告附带‘转储文件已经存在’信息的错误。

  ```
  gs_dump -p port_number postgres -f backup.sql -F plain --dont-overwrite-file
  ```

  说明

  - -s/--schema-only和-a/--data-only不能同时使用。
  - -c/--clean和-a/--data-only不能同时使用。
  - --inserts/--column-inserts和-o/--oids不能同时使用，因为INSERT命令不能设置OIDS。
  - --role和--rolepassword必须一起使用。
  - --binary-upgrade-usermap和--binary-upgrade必须一起使用。
  - --include-depend-objs/--exclude-self需要同-t/--include-table-file参数关联使用才会生效
  - --exclude-self必须同--include-depend-objs一起使用。

连接参数：

- -h, --host=HOSTNAME

  指定主机名称。如果数值以斜杠开头，则被用作到Unix域套接字的路径。缺省从PGHOST环境变量中获取（如果已设置），否则，尝试一个Unix域套接字连接。

  该参数只针对集群外，对集群内本机只能用127.0.0.1。

  例如：主机名

  环境变量：PGHOST

- -p, --port=PORT

  指定主机端口。

  环境变量：PGPORT

- -U, --username=NAME

  指定所连接主机的用户名。

  环境变量：PGUSER

- -w, --no-password

  不出现输入密码提示。如果主机要求密码认证并且密码没有通过其它形式给出，则连接尝试将会失败。 该选项在批量工作和不存在用户输入密码的脚本中很有帮助。

- -W, --password=PASSWORD

  指定用户连接的密码。如果主机的认证策略是trust，则不会对系统管理员进行密码验证，即无需输入-W选项；如果没有-W选项，并且不是系统管理员，“Dump Restore工具”会提示用户输入密码。

- --role=ROLENAME

  指定创建转储使用的角色名。选择该选项，会使gs_dump连接数据库后，发起一个SET ROLE角色名命令。当所授权用户（由-U指定）没有gs_dump要求的权限时，该选项会起到作用，即切换到具备相应权限的角色。某些安装操作规定不允

许直接以超系统管理员身份登录，而使用该选项能够在不违反该规定的情况下完成转储。

- --rolepassword=ROLEPASSWORD

  指定角色名的密码。

说明

场景1

如果某数据库集群有任何本地数据要添加到template1数据库，请谨慎将gs_dump的输出恢复到一个真正的空数据库中，否则可能会因为被添加对象的定义被复制，出现错误。要创建一个无本地添加的空数据库，需从template0而非template1复制，例如：

```
CREATE DATABASE foo WITH TEMPLATE template0;
```

tar归档形式的文件大小不得超过8GB（tar文件格式的固有限制）。tar文档整体大小和任何其他输出格式没有限制，操作系统可能对此有要求。

由gs_dump生成的转储文件不包含优化程序用来做执行计划决定的统计数据。因此，建议从某转储文件恢复之后运行ANALYZE以确保最佳效果。转储文件不包含任何ALTER DATABASE…SET命令，这些设置由gs_dumpall转储，还有数据库用户和其他完成安装设置。

场景2

当SEQUENCE已经到达最大或最小值时，通过gs_dump来备份SEQUENCE值会因执行报错退出。可参考如下说明处理：

1. SEQUENCE已经到达最大值，但最大值小于2^63-2

报错示例：

sequence对象定义

```
CREATE SEQUENCE seq INCREMENT 1 MINVALUE 1 MAXVALUE 3 START WITH 1;
```

执行gs_dump备份

```
gs_dump -U dbadmin -W {password} -p 37300 postgres -t PUBLIC.seq -f backup/MPPDB_backup.sql
gs_dump[port='37300'][postgres][2019-12-27 15:09:49]: The total objects number is 337.
gs_dump[port='37300'][postgres][2019-12-27 15:09:49]: WARNING:  get invalid xid from GTM because connection is not established
gs_dump[port='37300'][postgres][2019-12-27 15:09:49]: WARNING:  Failed to receive GTM rollback transaction response  for aborting prepared (null).
gs_dump: [port='37300'] [postgres] [archiver (db)] [2019-12-27 15:09:49] query failed: ERROR:  Can not connect to gtm when getting gxid, there is a connection error.
gs_dump: [port='37300'] [postgres] [archiver (db)] [2019-12-27 15:09:49] query was: RELEASE bfnextval
```

处理方法：

通过SQL语句连接postgres数据库，执行如下语句，修改sequence seq1的最大值。

```
gsql -p 37300 postgres -r -c "ALTER SEQUENCE PUBLIC.seq MAXVALUE 10;"
```

执行dump工具进行备份。

```
gs_dump -U dbadmin -W {password} -p 37300 postgres -t PUBLIC.seq -f backup/MPPDB_backup.sql
gs_dump[port='37300'][postgres][2019-12-27 15:10:53]: The total objects number is 337.
gs_dump[port='37300'][postgres][2019-12-27 15:10:53]: [100.00%] 337 objects have been dumped.
gs_dump[port='37300'][postgres][2019-12-27 15:10:53]: dump database postgres successfully
gs_dump[port='37300'][postgres][2019-12-27 15:10:53]: total time: 230  ms
```

2. SEQUENCE已经到达最小值或最大值2^63-2

gs_dump不支持该场景下的SEQUENCE数值备份。

说明

SQL端不支持SEQUENCE到达最大值2^63-2后的MAXVALUE修改，不支持SEQUENCE到达最小值后的MINVALUE修改。

场景3

gs_dump主要用于全库元数据导出场景，对导出单表做过性能优化，但是导出多表性能较差。对于导出多表场景，建议逐个表导出。例如：

```
gs_dump -U dbadmin -W {password} -p 37300 postgres -t public.table01 -s -f backup/table01.sql
gs_dump -U dbadmin -W {password} -p 37300 postgres -t public.table02 -s -f backup/table02.sql
```

如果业务停止情况下，或者业务空闲期，可以增加--non-lock-table参数提升gs_dump的性能。例如：

```
gs_dump -U dbadmin -W {password} -p 37300 postgres -t public.table03 -s --non-lock-table -f backup/
table03.sql
```

示例

使用gs_dump转储数据库为SQL文本文件或其它格式的操作，如下所示。

示例中“password”表示数据库用户密码，由用户自己设置；“backup/MPPDB_backup.sql”表示导出的文件，其中backup表示相对于当前目录的相对目录；“37300”表示数据库服务器端口；“postgres”表示要访问的数据库名。

说明

导出操作时，请确保该目录存在并且当前的操作系统用户对其具有读写权限。

示例1：执行gs_dump，导出postgres数据库全量信息，导出的MPPDB_backup.sql文件格式为纯文本格式。

```
gs_dump -U dbadmin -W {password} -f backup/MPPDB_backup.sql -p 37300 postgres -F p
gs_dump[port='37300'][postgres][2018-06-27 09:49:17]: The total objects number is 356.
gs_dump[port='37300'][postgres][2018-06-27 09:49:17]: [100.00%] 356 objects have been dumped.
gs_dump[port='37300'][postgres][2018-06-27 09:49:17]: dump database postgres successfully
gs_dump[port='37300'][postgres][2018-06-27 09:49:17]: total time: 1274  ms
```

使用gsql程序从纯文本导出文件中导入数据。

示例2：执行gs_dump，导出postgres数据库全量信息，导出的MPPDB_backup.tar文件格式为tar格式。

```
gs_dump -U dbadmin -W {password} -f backup/MPPDB_backup.tar -p 37300 postgres -F t
gs_dump[port='37300'][postgres][2018-06-27 10:02:24]: The total objects number is 1369.
gs_dump[port='37300'][postgres][2018-06-27 10:02:53]: [100.00%] 1369 objects have been dumped.
gs_dump[port='37300'][postgres][2018-06-27 10:02:53]: dump database postgres successfully
gs_dump[port='37300'][postgres][2018-06-27 10:02:53]: total time: 50086  ms
```

示例3：执行gs_dump，导出postgres数据库全量信息，导出的MPPDB_backup.dmp文件格式为自定义归档格式。

```
gs_dump -U dbadmin -W {password} -f backup/MPPDB_backup.dmp -p 37300 postgres -F c
gs_dump[port='37300'][postgres][2018-06-27 10:05:40]: The total objects number is 1369.
gs_dump[port='37300'][postgres][2018-06-27 10:06:03]: [100.00%] 1369 objects have been dumped.
gs_dump[port='37300'][postgres][2018-06-27 10:06:03]: dump database postgres successfully
gs_dump[port='37300'][postgres][2018-06-27 10:06:03]: total time: 36620  ms
```

示例4：执行gs_dump，导出postgres数据库全量信息，导出的MPPDB_backup文件格式为目录格式。

```
gs_dump -U dbadmin -W {password} -f backup/MPPDB_backup -p 37300  postgres -F d
gs_dump[port='37300'][postgres][2018-06-27 10:16:04]: The total objects number is 1369.
```

```
gs_dump[port='37300'][postgres][2018-06-27 10:16:23]: [100.00%] 1369 objects have been dumped.
gs_dump[port='37300'][postgres][2018-06-27 10:16:23]: dump database postgres successfully
gs_dump[port='37300'][postgres][2018-06-27 10:16:23]: total time: 33977  ms
```

示例5：执行gs_dump，导出postgres数据库信息，但不导出/home/MPPDB_temp.sql中指定的表信息。导出的MPPDB_backup.sql文件格式为纯文本格式。

```
gs_dump -U dbadmin -W {password} -p 37300 postgres --exclude-table-file=/home/MPPDB_temp.sql -f
backup/MPPDB_backup.sql
gs_dump[port='37300'][postgres][2018-06-27 10:37:01]: The total objects number is 1367.
gs_dump[port='37300'][postgres][2018-06-27 10:37:22]: [100.00%] 1367 objects have been dumped.
gs_dump[port='37300'][postgres][2018-06-27 10:37:22]: dump database postgres successfully
gs_dump[port='37300'][postgres][2018-06-27 10:37:22]: total time: 37017  ms
```

示例6：执行gs_dump，仅导出依赖于指定表testtable的视图信息。然后创建新的testtable表，再恢复依赖其上的视图。

备份仅依赖于testtable的视图

```
gs_dump -s -p 37300 postgres -t PUBLIC.testtable --include-depend-objs --exclude-self -f backup/
MPPDB_backup.sql -F p
gs_dump[port='37300'][postgres][2018-06-15 14:12:54]: The total objects number is 331.
gs_dump[port='37300'][postgres][2018-06-15 14:12:54]: [100.00%] 331 objects have been dumped.
gs_dump[port='37300'][postgres][2018-06-15 14:12:54]: dump database postgres successfully
gs_dump[port='37300'][postgres][2018-06-15 14:12:54]: total time: 327  ms
```

修改testtable名称

```
gsql -p 37300 postgres -r -c "ALTER TABLE PUBLIC.testtable RENAME TO testtable_bak;"
```

创建新的testtable表

```
CREATE TABLE PUBLIC.testtable(a int, b int, c int);
```

还原依赖于testtable的视图

```
gsql -p 37300 postgres -r -f backup/MPPDB_backup.sql
```

相关命令

gs_dumpall，**gs_restore**

7.2 gs_dumpall

背景信息

gs_dumpall是DWS用于导出所有数据库相关信息工具，它可以导出集群数据库的所有数据，包括默认数据库postgres的数据、自定义数据库的数据、以及集群所有数据库公共的全局对象。

gs_dumpall工具在进行数据导出时，导出的表会被加锁，会导致阻塞读写。

gs_dumpall工具支持导出完整一致的数据。例如，T1时刻启动gs_dumpall导出整个集群数据库，那么导出数据结果将会是T1时刻该集群数据库的数据状态，T1时刻之后对集群数据库的修改不会被导出。

gs_dumpall在导出整个集群所有数据库时分为两部分：

- gs_dumpall自身对所有数据库公共的全局对象进行导出，包括有关数据库用户和组，表空间以及属性（例如，适用于数据库整体的访问权限）信息。
- gs_dumpall通过调用gs_dump来完成集群中各数据库的SQL脚本文件导出，该脚本文件包含将数据库恢复为其保存时的状态所需要的全部SQL语句。

以上两部分导出的结果为纯文本格式的SQL脚本文件，使用gsql运行该脚本文件可以恢复集群数据库。

注意事项

- 禁止修改导出的文件和内容，否则可能无法恢复成功。
- 为了保证数据一致性和完整性，gs_dumpall会对需要转储的表设置共享锁。如果某张表在别的事务中设置了共享锁，gs_dumpall会等待此表的锁释放后锁定此表。如果无法在指定时间内锁定某张表，转储会失败。用户可以通过指定--lock-wait-timeout选项，自定义等待锁超时时间。
- 由于gs_dumpall读取所有数据库中的表，因此必须以数据库集群管理员身份进行连接，才能导出完整文件。在使用gsql执行脚本文件导入时，同样需要管理员权限，以便添加用户和组，以及创建数据库。

语法

gs_dumpall [OPTION]...

参数说明

通用参数：

- -f, --filename=FILENAME

  将输出发送至指定文件。如果这里省略，则使用标准输出。

- -v, --verbose

  指定verbose模式。该选项将导致gs_dumpall向转储文件输出详细的对象注解和启动/停止次数，向标准错误流输出处理信息。

- -V, --version

  打印gs_dumpall版本，然后退出。

- --lock-wait-timeout=TIMEOUT

  请勿在转储刚开始时一直等待以获取共享表锁。如果无法在指定时间内锁定某个表，就选择失败。可以以任何符合SET statement_timeout的格式指定超时时间。

- -?, --help

  显示gs_dumpall命令行参数帮助，然后退出。

转储参数：

- -a, --data-only

  只转储数据，不转储模式（数据定义）。

- -c, --clean

  在重新创建数据库之前，执行SQL语句清理（删除）这些数据库。针对角色和表空间的转储命令已添加。

- -g, --globals-only

  只转储全局对象（角色和表空间），无数据库。

- -o, --oids

  转储每个表的对象标识符（OIDs），作为表的一部分数据。该选项用于应用以某种方式（例如：外键约束方式）参照了OID列的情况。如果不是以上这种情况，请勿使用该选项。

- -O, --no-owner

  不输出设置对象的归属这样的命令，以匹配原始数据库。默认情况下，gs_dumpall会发出ALTER OWNER或SET SESSION AUTHORIZATION语句设置所创建的模式元素的所属。如果脚本正在运行，该语句不会执行成功，除非是由系统管理员触发（或是拥有脚本中所有对象的同一个用户）。通过指定-O，编写一个任何用户都能存储的脚本，且该脚本会授予该用户拥有所有对象的权限。

- -r, --roles-only

  只转储角色，不转储数据库或表空间。

- -s, --schema-only

  只转储对象定义（模式），而非数据。

- -S, --sysadmin=NAME

  在转储过程中使用的系统管理员名称。

- -t, --tablespaces-only

  只转储表空间，不转储数据库或角色。

- -x, --no-privileges

  防止转储访问权限（授权/撤销命令）。

- --column-inserts|--attribute-inserts

  以INSERT命令带列名（INSERT INTO表（列、…）值…）方式导出数据。这会导致恢复缓慢。但是由于该选项会针对每行生成一个独立分开的命令，所以在重新加载某行时出现的错误只会导致那行丢失，而非整个表内容。

- --disable-dollar-quoting

  该选项将禁止在函数体前使用美元符号$，并强制使用SQL标准字符串语法对其进行引用。

- --disable-triggers

  该参数为扩展预留接口，不建议使用。

- --inserts

  发出INSERT命令（而非COPY命令）时转储数据。这会导致恢复缓慢。注意如果重排列顺序，可能会导致恢复整个失败。--column-inserts选项更加安全，虽然可能更慢些。

- --no-security-labels

  该参数为扩展预留接口，不建议使用。

- --no-tablespaces

  该参数在8.2.0.100版本中已废弃，为兼容历史版本功能保留该函数。

  请勿输出创建表空间的命令，也请勿针对对象选择表空间。使用该选项，无论默认表空间是哪个，在恢复过程中所有对象都会被创建。

- --no-unlogged-table-data

  该参数为扩展预留接口，不建议使用。

- --quote-all-identifiers

  强制对所有标识符加引号。为了向后续版本迁移，且其中可能涉及引入额外关键词，在转储相应数据库时该选项会有帮助。

- --dont-overwrite-file

  不重写当前文件。

- --use-set-session-authorization
  输出符合SQL标准的SET SESSION AUTHORIZATION命令而不是ALTER OWNER命令来确定对象所有权。这样令转储更加符合标准，但是如果转储文件中的对象的历史有些问题，那么可能不能正确恢复。并且，使用SET SESSION AUTHORIZATION的转储需要数据库系统管理员的权限才能转储成功，而ALTER OWNER需要的权限则低得多。
- --with-encryption=AES128
  指定转储数据需用AES128进行加密。
- --with-key=KEY
  AES128密钥长度必须是16字节。
- --include-extensions
  如果include-extensions参数被设置，将备份所有的CREATE EXTENSION语句。
- --include-templatedb
  转储过程中包含模板库。
- --dump-nodes
  转储过程中包含节点和Node Group。
- --include-nodes
  将TO NODE语句包含在已转储的CREATE TABLE命令中。
- --include-buckets
  该参数为扩展预留接口，不建议使用。
- --dump-wrm
  存储过程中包含负载资源管理器，具体包括资源池、负载组以及负载组映射。
- --binary-upgrade
  该参数为扩展预留接口，不建议使用。
- --binary-upgrade-usermap="USER1=USER2"
  该参数为扩展预留接口，不建议使用。
- --tablespaces-postfix
  该参数为扩展预留接口，不建议使用。
- --parallel-jobs
  指定备份进程并发数，取值范围为1~1000。

说明

- -g/--globals-only和-r/--roles-only不能同时使用。
- -g/--globals-only和-t/--tablespaces-only不能同时使用。
- -r/--roles-only和-t/--tablespaces-only不能同时使用。
- -s/--schema-only和-a/--data-only不能同时使用。
- -r/--roles-only和-a/--data-only不能同时使用。
- -t/--tablespaces-only和-a/--data-only不能同时使用。
- -g/--globals-only和-a/--data-only不能同时使用。
- --tablespaces-postfix和--binary-upgrade必须一起使用。
- --parallel-jobs和-f/--file必须一起使用。

连接参数：

- -h, --host

  指定主机的名称。如果取值是以斜线开头，它将用作Unix域套接字的目录。默认值取自PGHOST环境变量；如果没有设置，将启动某个Unix域套接字建立连接。

  该参数只针对集群外，对集群内本机只能用127.0.0.1。

  环境变量：PGHOST

- -l, --database

  指定所连接的转储全局对象的数据库名称，并去寻找还有其他哪些数据库需要被转储。如果没有指定，会使用postgres数据库，如果postgres数据库不存在，会使用template1。

- -p, --port

  指定服务器所监听的TCP端口或本地Unix域套接字后缀，以确保连接。默认值设置为PGPORT环境变量。

  环境变量：PGPORT

- -U, --username

  所连接的用户名。

  环境变量：PGUSER

- -w, --no-password

  不出现输入密码提示。如果服务器要求密码认证并且密码没有通过其它形式给出，则连接尝试将会失败。 该选项在批量工作和不存在用户输入密码的脚本中很有帮助。

- -W, --password

  指定用户连接的密码。如果主机的认证策略是trust，则不会对系统管理员进行密码验证，即无需输入-W选项；如果没有-W选项，并且不是系统管理员，“Dump Restore工具”会提示用户输入密码。

- --role

  指定创建转储使用的角色名。选择该选项，会使gs_dumpall连接数据库后，发起一个SET ROLE角色名命令。当所授权用户（由-U指定）没有gs_dumpall要求的权限时，该选项会起到作用，即切换到具备相应权限的角色。某些安装操作规定不允许直接以系统管理员身份登录，而使用该选项能够在不违反该规定的情况下完成转储。

- --rolepassword

  指定具体角色用户的角色密码。

说明

由于gs_dumpall内部调用**gs_dump**，所以一些诊断信息参见**gs_dump**。

一旦恢复，建议在每个数据库上运行ANALYZE，优化程序提供有用的统计数据。

gs_dumpall恢复前需要所有必要的表空间目录才能退出；否则，对于处在非默认位置的数据库，数据库创建会失败。

示例

使用gs_dumpall一次导出集群的所有数据库。

说明

gs_dumpall仅支持纯文本格式导出。所以只能使用gsql恢复gs_dumpall导出的转储内容。

```
gs_dumpall -f backup/bkp2.sql -p 37300
gs_dump[port='37300'][dbname='postgres'][2018-06-27 09:55:09]: The total objects number is 2371.
gs_dump[port='37300'][dbname='postgres'][2018-06-27 09:55:35]: [100.00%] 2371 objects have been dumped.
gs_dump[port='37300'][dbname='postgres'][2018-06-27 09:55:46]: dump database dbname='postgres' successfully
gs_dump[port='37300'][dbname='postgres'][2018-06-27 09:55:46]: total time: 55567  ms
gs_dumpall[port='37300'][2018-06-27 09:55:46]: dumpall operation successful
gs_dumpall[port='37300'][2018-06-27 09:55:46]: total time: 56088  ms
```

相关命令

gs_dump，**gs_restore**

7.3 gs_restore

背景信息

gs_restore是DWS提供的针对gs_dump导出数据的导入工具。通过此工具可由gs_dump生成的导出文件进行导入。

主要功能包含：

- 导入到数据库

  如果连接参数中指定了数据库，则数据将被导入到指定的数据库中。其中，并行导入必须指定连接的密码。
- 导入到脚本文件

  如果未指定导入数据库，则创建包含重建数据库所必须的SQL语句脚本并写入到文件或者标准输出。等效于直接使用gs_dump导出为纯文本格式。

命令格式

```
gs_restore [OPTION]... FILE
```

说明

- FILE没有短选项或长选项。用来指定归档文件所处的位置。
- 作为前提条件，需输入dbname或-l选项。不允许用户同时输入dbname和-l选项。
- gs_restore默认是以追加的方式进行数据导入。为避免多次导入造成数据异常，在进行导入时，建议使用"-e"和"-c" 参数，即导入前删除已存在于待导入数据库中的数据库对象，同时当出现导入错误时，忽略当前错误，继续执行导入任务，并在导入后会显示相应的错误信息。

参数说明

通用参数：

- -d, --dbname=NAME

  连接数据库dbname并直接导入到该数据库中。
- -f, --file=FILENAME

  指定生成脚本的输出文件，或使用-l时列表的输出文件。

默认是标准输出。

说明

-f不能同-d一起使用。

- -F, --format=c|d|t

  指定归档格式。由于gs_restore会自动决定格式，因此不需要指定格式。

  取值范围：

  - c/custom：该归档形式为4.21-gs_dump的自定义格式。
  - d/directory：该归档形式是一个目录归档形式。
  - t/tar：该归档形式是一个tar归档形式。

- -l, --list

  列出归档形式内容。这一操作的输出可用作-L选项的输入。注意如果像-n或-t的过滤选项与-l使用，过滤选项将会限制列举的项目（即归档形式内容）。

- -v, --verbose

  指定verbose模式。

- -V, --version

  打印gs_restore版本，然后退出。

- -?, --help

  显示gs_restore命令行参数帮助，然后退出。

导入参数：

- -a, -data-only

  只导入数据，不导入模式（数据定义）。gs_restore的导入是以追加方式进行的。

- -c, --clean

  在重新创建数据库对象前，清理（删除）已存在于将要还原的数据库中的数据库对象

- -C, --create

  导入到数据库之前请创建数据库。（选择该选项后，以-d打头的数据库将被用作发布首个CREATE DATABASE命令。所有数据将被导入到出现在归档文件的数据库中。）

- -e, --exit-on-error

  当发送SQL语句到数据库时如果出现错误，请退出。默认状态下会继续，且在导入后会显示一系列错误信息。

- -I, --index=NAME

  只导入已列举的index的定义。允许导入多个index。如果多次输入-I index导入多个index。

  例如：

  ```
  gs_restore -h host_name -p port_number -d gaussdb -I Index1 -I Index2 backup/MPPDB_backup.tar
  ```

  在上面这个例子中，Index1和Index2会被导入。

- -j, --jobs=NUM

  运行gs_restore最耗时的部分（如加载数据、创建index、或创建约束）使用并发任务。该选项能大幅缩短导入时间，即将一个大型数据库导入到某一多处理器的服务器上。

每个任务可能是一个进程或一个线程，这由操作系统决定；每个任务与服务器进行单独连接。

该选项的最优值取决于服务器的硬件设置、客户端、以及网络。还包括这些因素，如CPU核数量、硬盘设置。建议是从增加服务器上的CPU核数量入手，更大的值（服务器上CPU核数量）在很多情况下也能导致数据文件更快的被导入。需要注意，过高的值会由于超负荷反而导致性能降低。

该选项只支持自定义归档格式。输入文件必须是常规文件（不能是像pipe的文件）。如果是通过脚本文件，而非直接连接数据库服务器，该选项可忽略。而且，多任务不能与--single-transaction选项一起使用。

- -L, --use-list=FILENAME

  只导入列举在list-file中的那些归档形式元素，导入顺序以它们在文件中的顺序为准。注意如果像-n或-t的过滤选项与-L使用，它们将会进一步限制导入的项目。

  一般情况下，list-file是通过编辑前面提到的某个-l参数的输出创建的。文件行的位置可更改或直接删除行，也可使用分号（;）在行的开始注出。见下文的举例。

- -n, --schema=NAME

  只导入已列举的模式中的对象。

  该选项可与-t选项一起用于导入某个指定的表。

  多次输入-n *schemaname*可以导入多个模式。

  例如：

  ```
  gs_restore -h host_name -p port_number -d gaussdb -n sch1 -n sch2 backup/MPPDB_backup.tar
  ```

  在上面这个例子中，sch1和sch2会被导入。

- -O, --no-owner

  不输出设置对象的归属这样的命令，以匹配原始数据库。默认情况下，gs_restore会发出ALTER OWNER或SET SESSION AUTHORIZATION语句设置所创建的模式元素的所属。除非是由系统管理员（或是拥有脚本中所有对象的同一个用户）进行数据库首次连接的操作，否则语句会失败。使用-O选项，任何用户名都可用于首次连接，且该用户拥有所有已创建的对象。

- -P, --function=NAME(args)

  只导入已列举的函数。请按照函数所在转储文件中的目录，准确拼写函数名称和参数。

  当-P单独使用时，表示导入文件中所有'function-name(args)'函数；当-P同-n一起使用时，表示导入指定模式下的'function-name(args)'函数；多次输入-P，而仅指定一次-n，表示所有导入的函数默认都是位于-n模式下的。

  可以多次输入-n schema-name -P 'function-name(args)'同时导入多个指定模式下的函数。

  例如：

  ```
  ./gs_restore -h host_name -p port_number -d gaussdb -n test1 -P 'Func1(integer)' -n test2 -P 'Func2(integer)' backup/MPPDB_backup.tar
  ```

  在上面这个例子中，test1模式下的函数Func1(i integer)和test2模式下的函数Func2(j integer)会被一起导入。

- -s, --schema-only

  只导入模式（数据定义），不导入数据（表内容）。当前的序列值也不会导入。

- -S, --sysadmin=NAME

  该参数为扩展预留接口，不建议使用。

- -t, --table=NAME

只导入已列举的表定义、数据或定义和数据。该选项与-n选项同时使用时，用来指定某个模式下的表对象。-n参数不输入时，默认为PUBLIC模式。多次输入-n <schemaname> -t <tablename>可以导入指定模式下的多个表。

例如：

导入PUBLIC模式下的table1

```
gs_restore -h host_name -p port_number -d gaussdb -t table1 backup/MPPDB_backup.tar
```

导入test1模式下的test1和test2模式下test2

```
gs_restore -h host_name -p port_number -d gaussdb -n test1 -t test1 -n test2 -t test2 backup/MPPDB_backup.tar
```

导入PUBLIC模式下的table1和test1 模式下test1

```
gs_restore -h host_name -p port_number -d gaussdb -n PUBLIC -t table1 -n test1 -t table1 backup/MPPDB_backup.tar
```

须知

-t不支持schema_name.table_name的输入格式。

- -T, --trigger=NAME

  该参数为扩展预留接口。

- -x, --no-privileges/--no-acl

  防止导入访问权限（grant/revoke命令）。

- -1, --single-transaction

  执行导入作为一个单独事务（即把命令包围在BEGIN/COMMIT中）。

  该选项确保要么所有命令成功完成，要么没有改变应用。该选项意为--exit-on-error。

- --disable-triggers

  该参数为扩展预留接口，不建议使用。

- --no-data-for-failed-tables

  默认状态下，即使创建表的命令失败（如表已经存在），表数据仍会被导入。使用该选项，像这种表的数据会被跳过。如果目标数据库已包含想要的表内容，这种行为会有帮助。

  该选项只有在直接导入到某数据库中时有效，不针对生成SQL脚本文件输出。

- --no-security-labels

  该参数为扩展预留接口，不建议使用。

- --no-tablespaces

  该参数在8.2.0.100版本中已废弃，为兼容历史版本功能保留该函数。

  不输出选择表空间的命令。使用该选项，无论默认表空间是哪个，在导入过程中所有对象都会被创建。

- --section=SECTION

  导入已列举的区段（如pre-data、data、或post-data）。

- --use-set-session-authorization

  该选项用来进行文本格式的备份。

  输出SET SESSION AUTHORIZATION命令，而非ALTER OWNER命令，用于决定对象归属。该选项使转储更加兼容标准，但通过参考转储中对象的记录，导入过

程可能会有问题。使用SET SESSION AUTHORIZATION的转储要求必须是系统管理员，同时在导入前还需参考"SET SESSION AUTHORIZATION"，手工对导出文件的密码进行修改验证，只有这样才能进行正确的导入操作，相比之下，ALTER OWNER对权限要求较低。

- --with-key=KEY

  AES128密钥长度必须是16字节。

  说明

  如果转储被加密，则必须在gs_restore命令中输入--with-key <keyname>选项。如果未输入，用户会收到错误信息。

  应该输入转储时所输入的相同的key。

须知

- 如果安装过程中有任何本地数据要添加到template1数据库，请谨慎将gs_restore的输出载入到一个真正的空数据库中；否则可能会因为被添加对象的定义被复制，而出现错误。要创建一个无本地添加的空数据库，需从template0而非template1复制，例如：

```
CREATE DATABASE foo WITH TEMPLATE template0;
```

- gs_restore不能选择性地导入大对象；例如只能导入那些指定表的对象。如果某个归档形式包含大对象，那所有大对象都会被导入，或一个都不会被导入，如果它们通过-L、-t或其他选项被排除。

说明

- -d/--dbname 和 -f/--file 不能同时使用；
- -s/--schema-only 和 -a/--data-only不能同时使用；
- -c/--clean 和 -a/--data-only不能同时使用；
- 使用--single-transaction时，-j/--jobs必须为单任务；
- --role 和 --rolepassword必须一起使用。

连接参数：

- -h, --host=HOSTNAME

  指定的主机名称。如果取值是以斜线开头，他将用作Unix域套接字的目录。默认值取自PGHOST环境变量；如果没有设置，将启动某个Unix域套接字建立连接。

  该参数只针对集群外，对集群内本机只能用127.0.0.1。

- -p, --port=PORT

  指定服务器所监听的TCP端口或本地Unix域套接字后缀，以确保连接。默认值设置为PGPORT环境变量。

- -U, --username=NAME

  所连接的用户名。

- -w, --no-password

  不出现输入密码提示。如果服务器要求密码认证并且密码没有通过其它形式给出，则连接尝试将会失败。 该选项在批量工作和不存在用户输入密码的脚本中很有帮助。

- -W, --password=PASSWORD

指定用户连接的密码。如果主机的认证策略是trust，则不会对系统管理员进行密码验证，即无需输入-W参数；如果没有-W参数，并且不是系统管理员，“gs_restore”会提示用户输入密码。

- --role=ROLENAME

  指定导入操作使用的角色名。选择该参数，会使gs_restore连接数据库后，发起一个SET ROLE角色名命令。当所授权用户（由-U指定）没有gs_restore要求的权限时，该参数会起到作用，即切换到具备相应权限的角色。某些安装操作规定不允许直接以初始用户身份登录，而使用该参数能够在不违反该规定的情况下完成导入。

- --rolepassword=ROLEPASSWORD

  指定具体角色用户的角色密码。

示例

特例：执行gsql程序，使用如下选项导入由gs_dump/gs_dumpall生成导出文件夹（纯文本格式）的MPPDB_backup.sql文件到gaussdb数据库。

```
gsql -d gaussdb -p 8000 -W {password} -f /home/omm/test/MPPDB_backup.sql
SET
SET
SET
SET
SET
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
CREATE INDEX
CREATE INDEX
CREATE INDEX
SET
CREATE INDEX
REVOKE
REVOKE
GRANT
GRANT
total time: 30476  ms
```

gs_restore用来导入由gs_dump生成的导出文件。

示例1：执行gs_restore，将导出的MPPDB_backup.dmp文件（自定义归档格式）导入到gaussdb数据库。

```
gs_restore -W {password} backup/MPPDB_backup.dmp -p 8000 -d gaussdb
gs_restore: restore operation successful
gs_restore: total time: 13053  ms
```

示例2：执行gs_restore，将导出的MPPDB_backup.tar文件（tar格式）导入到gaussdb数据库。

```
gs_restore backup/MPPDB_backup.tar -p 8000 -d gaussdb
gs_restore[2017-07-21 19:16:26]: restore operation successful
gs_restore[2017-07-21 19:16:26]: total time: 21203  ms
```

示例3：执行gs_restore，将导出的MPPDB_backup文件（目录格式）导入到gaussdb数据库。

```
gs_restore backup/MPPDB_backup -p 8000 -d gaussdb
gs_restore[2017-07-21 19:16:26]: restore operation successful
gs_restore[2017-07-21 19:16:26]: total time: 21003  ms
```

示例4：执行gs_restore，使用自定义归档格式的MPPDB_backup.dmp文件来进行如下导入操作。 导入PUBLIC模式下所有对象的定义和数据。在导入时会先删除已经存在的对象，如果原对象存在跨模式的依赖则需手工强制干预。

```
gs_restore backup/MPPDB_backup.dmp -p 8000 -d gaussdb -e -c -n PUBLIC
gs_restore: [archiver (db)] Error while PROCESSING TOC:
gs_restore: [archiver (db)] Error from TOC entry 313; 1259 337399 TABLE table1 gaussdba
gs_restore: [archiver (db)] could not execute query: ERROR:  cannot drop table table1 because other objects depend on it
DETAIL:  view t1.v1 depends on table table1
HINT:  Use DROP ... CASCADE to drop the dependent objects too.
   Command was: DROP TABLE public.table1;
```

手工删除依赖，导入完成后再重新创建。

```
gs_restore backup/MPPDB_backup.dmp -p 8000 -d gaussdb -e -c -n PUBLIC
gs_restore[2017-07-21 19:16:26]: restore operation successful
gs_restore[2017-07-21 19:16:26]: total time: 2203  ms
```

示例5：执行gs_restore，使用自定义归档格式的MPPDB_backup.dmp文件来进行如下导入操作。只导入PUBLIC模式下表table1的定义。

```
gs_restore backup/MPPDB_backup.dmp -p 8000 -d gaussdb -e -c -s -n PUBLIC -t table1
gs_restore[2017-07-21 19:16:26]: restore operation successful
gs_restore[2017-07-21 19:16:26]: total time: 21000  ms
```

示例6：执行gs_restore，使用自定义归档格式的MPPDB_backup.dmp文件来进行如下导入操作。只导入PUBLIC模式下表table1的数据。

```
gs_restore backup/MPPDB_backup.dmp -p 8000 -d gaussdb -e -a -n PUBLIC -t table1
gs_restore[2017-07-21 19:16:26]: restore operation successful
gs_restore[2017-07-21 19:16:26]: total time: 20203  ms
```

说明

创建集群的时候会启动调度器，启动调度器时会创建调度器的一些资源，包括调度器的表所在的schema scheduler，调度器运行时创建的几张表bandwidth_history_table,cpu_template_storage,io_template_storage,mem_template_storage,scheduler_config,scheduler_storage,task_history_storage,task_storage,vacuum_full_rslt,function scheduler_workload_query_func,pg_task在执行gs_restore的时候，会将调度器的表、schema和索引等对象也一起恢复，由于调度器是一个常驻进程，新建的集群也会自动的创建这些对象，所以在执行gs_restore时会发生调度器的对象存在的错误信息，该报错对集群正常操作没有影响，可忽略。

相关命令

gs_dump，gs_dumpall

7.4 gds_check

背景信息

gds_check用于对GDS部署环境进行检查，包括操作系统参数、网络环境、磁盘占用情况等，也支持对可修复系统参数的修复校正，有助于在部署运行GDS时提前发现潜在问题，提高执行成功率。

注意事项

- 执行脚本前需设置环境变量，可参考章节。

- 脚本需要在python 3环境下运行。
- 必须在root用户下执行脚本。
- 必须指定-t、--host参数。
- 当--host指定网络地址0.0.0.0或127.0.0.1时，不会检查MTU和网卡多队列。
- 网卡多队列的检查、修复要求网卡至少是万兆。
- --host参数指定的所有节点的密码必须保持一致，才能保证脚本成功进行远程检查。
- 执行修复时，对配置劣于推荐值的参数，建议设置为OS中配置项的推荐值，具体见下表：

表 7-2 OS 配置项

参数	推荐值
net.core.somaxconn	65535
net.ipv4.tcp_max_syn_backlog	65535
net.core.netdev_max_backlog	65535
net.ipv4.tcp_retries1	5
net.ipv4.tcp_retries2	12
net.ipv4.ip_local_port_range	26000~65535
MTU	1500
net.core.wmem_max	21299200
net.core.rmem_max	21299200
net.core.wmem_default	21299200
net.core.rmem_default	21299200
max handler	1000000
vm.swappiness	10

表 7-3 磁盘检查

检查项	警告
磁盘空间使用率	大于等于70%且小于90%
inode使用率	大于等于70%且小于90%

表 7-4 网络检查

检查项	报错
检查网络连通性	包100%丢失
检查网卡多队列	开启网卡多队列且绑定不同CPU，支持fix修改

语法

- 检查命令
  ```
  gds_check -t check --host [/path/to/hostfile | ipaddr1,ipaddr2...] --ping-host [/path/to/pinghostfile | ipaddr1,ipaddr2...] [--detail]
  ```
- 修复命令
  ```
  gds_check -t fix --host [/path/to/hostfile | ipaddr1,ipaddr2...] [--detail]
  ```

参数说明

- -t

  操作类型，表示检查/修复。

  取值：check，fix。
- --host

  需要检查/修复的节点IP列表。

  取值：IP列表，支持文件和字符串两种形式。
  - 文件形式：每一行一个IP地址，如：

    192.168.1.200

    192.168.1.201
  - 字符串形式：半角逗号分隔，如：

    192.168.1.200,192.168.1.201
- --ping-host

  在各检查节点上进行网络ping检查的目标地址。

  取值：IP列表，支持文件和字符串两种形式，一般是DN、CN、网关的IP地址。
  - 文件形式：每一行一个IP地址，如：

    192.168.2.200

    192.168.2.201
  - 字符串形式：半角逗号分隔，如：

    192.168.2.200,192.168.2.201
- --detail

  显示检查/修复项详细信息，并存入日志。
- -V

  显示版本信息。
- -h, --help

  显示帮助信息。

示例

执行检查，--host、--ping-host均为IP字符串形式：

```
gds_check -t check --host 192.168.1.100,192.168.1.101 --ping-host 192.168.2.100
```

执行检查，--host为字符串，--ping-host为文件形式：

```
gds_check -t check --host 192.168.1.100,192.168.1.101 --ping-host /home/gds/iplist

cat /home/gds/iplist
192.168.2.100
192.168.2.101
```

执行检查，--host为文件形式，--ping-host为字符串：

```
gds_check -t check --host  /home/gds/iplist --ping-host 192.168.1.100,192.168.1.101
```

执行修复，--host为字符串：

```
gds_check -t fix --host 192.168.1.100,192.168.1.101
```

执行检查，打印详细信息，并存入日志：

```
gds_check -t check --host 192.168.1.100 --detail
```

执行修复，打印详细信息，并存入日志：

```
gds_check -t fix --host 192.168.1.100 --detail
```

7.5 gds_install

背景信息

gds_install是用于批量安装gds的脚本工具，可大大提高GDS部署效率。

注意事项

- 执行脚本前需设置环境变量，可参考章节。
- 脚本需要在python 3环境下运行。
- 必须在root用户下执行脚本gds_install。
- 用户需要检查上层目录权限，保证GDS用户对安装操作目录、安装目录及安装包有读写执行的权限。
- 目前不支持跨平台的安装部署。
- 执行命令节点也必须是安装部署机器之一。
- --host参数指定的所有节点的密码必须保持一致，才能保证脚本成功进行远程部署。

语法

```
gds_install -I /path/to/install_dir -U user -G user_group --pkg /path/to/pkg.tar.gz --host [/path/to/hostfile | ipaddr1,ipaddr2...] [--ping-host [/path/to/hostfile | ipaddr1,ipaddr2...]]
```

参数说明

- -I
  安装目录。

默认值：/opt/${gds_user}/packages/，其中${gds_user}表示GDS业务的操作系统用户。

- -U
  GDS用户。
- -G
  GDS用户所属组。
- --pkg
  GDS安装包路径，形如/path/to/GaussDB-9.1.0-REDHAT-x86_64bit-Gds.tar.gz。
- --host
  待安装部署节点的IP列表，支持文件和字符串两种形式：
  - 文件形式：每一行一个IP地址，如：
    192.168.2.200
    192.168.2.201
  - 字符串形式：半角逗号分隔，如：
    192.168.2.200,192.168.2.201。

  说明

  执行命令节点必须是待部署节点之一，其IP须在列表中。
- --ping-host
  调用gds_check时，在各检查节点上进行网络ping检查的目标地址。
  取值：IP列表，支持文件和字符串两种形式，一般是DN、CN、网关的IP地址。
  - 文件形式：每一行一个IP地址，如：
    192.168.2.200
    192.168.2.201
  - 字符串形式：半角逗号分隔，如：
    192.168.2.200,192.168.2.201
- -V
  显示版本信息。
- -h, --help
  显示帮助信息。

示例

将GDS安装部署在节点192.168.1.100、192.168.1.101上，并指定安装目录为/opt/gdspackages/install_dir，GDS用户是gds_test:wheel。

```
gds_install -I /opt/gdspackages/install_dir --host 192.168.1.100,192.168.1.101 -U gds_test -G wheel --pkg /home/gds_test/GaussDB-9.1.0-REDHAT-x86_64bit-Gds.tar.gz
```

7.6 gds_uninstall

背景信息

gds_uninstall是用于批量卸载GDS的脚本工具。

注意事项

- 执行脚本前需设置环境变量，可参考章节。
- 脚本需要在python 3环境下运行。
- 必须在root用户下执行脚本gds_uninstall。
- 必须包含--host、-U参数。
- 目前不支持跨平台的卸载操作。
- --host参数指定的所有节点的密码必须保持一致，才能保证脚本成功进行远程卸载。

语法

```
gds_uninstall --host [/path/to/hostfile | ipaddr1,ipaddr2...] –U gds_user [--delete-user | --delete-user-and-group]
```

参数说明

- --host

  待卸载节点的IP列表，支持文件和字符串两种形式：
  - 文件形式：每一行一个IP地址，如：

    192.168.2.200

    192.168.2.201
  - 字符串形式：半角逗号分隔，如：

    192.168.2.200,192.168.2.201。
- -U

  GDS用户。
- --delete-user

  卸载的同时，删除用户。被删除的用户不可以是root用户。
- --delete-user-and-group

  卸载的同时，删除用户和其所在用户组。仅当用户组只包含该待删除用户一个用户时可以删除用户组。该用户组不能是root用户组。
- -V

  显示版本信息。
- -h, --help

  显示帮助信息。

示例

卸载安装部署在节点192.168.1.100、192.168.1.101上，安装用户为gds_test的，GDS文件夹及环境变量。

```
gds_uninstall -U gds_test --host 192.168.1.100,192.168.1.101
```

卸载时，同时删除用户。

```
gds_uninstall -U gds_test --host 192.168.1.100,192.168.1.101 --delete-user
```

卸载时，同时删除用户和用户组。

```
gds_uninstall -U gds_test --host 192.168.1.100,192.168.1.101 --delete-user-and-group
```

7.7 gds_ctl

背景信息

gds_ctl是一个批量控制GDS启停的脚本工具，一次执行可以在多个节点上启动/停止相同端口的GDS服务进程，并在启动时为每一个进程设置看护程序，用于看护GDS进程。

注意事项

- 执行脚本前需切换到GDS用户，必须在普通用户下执行脚本gds_ctl。
- 脚本需要在python 3环境下运行。
- gds_ctl继承了GDS主要命令行参数，除-p以及-h外，其他参数意义不变。在gds_ctl中，-p只需指定端口即可。
- 使用gds_ctl批量操作的节点必须是此前使用gds_install安装部署的节点。

语法

- 启动命令
  ```
  gds_ctl start --host [/path/to/hostfile | ipaddr1,ipaddr2...] -p PORT -d DATADIR -H ALLOW_IPs [gds other original options]
  ```
- 停止命令
  ```
  gds_ctl stop --host [/path/to/hostfile | ipaddr1,ipaddr2...] -p PORT
  ```
- 重启命令
  ```
  gds_ctl restart --host [/path/to/hostfile | ipaddr1,ipaddr2...] -p PORT
  ```

参数说明

- --host

  待运行GDS节点的IP列表，支持文件和字符串两种形式：
  - 文件形式：每一行一个IP地址，如：

    192.168.2.200

    192.168.2.201
  - 字符串形式：半角逗号分隔，如：

    192.168.2.200,192.168.2.201
- -p

  监听端口。

  取值范围：1024~65535，正整数。

  默认值：8098
- --help

  显示帮助信息。
- -V

  显示版本信息。

兼容 GDS 原参数

- -d dir

  设置待导入数据文件的目录。在GDS进程权限允许的条件下，-d指定的目录会自动被创建。

- -l log_file

  设置日志文件。

  与-R参数一起使用，可支持日志自动切分。当设置-R参数后，GDS会根据设置的值重新生成新的文件，以此来避免单个日志文件过大。

  生成规则：GDS默认只识别后缀是log的文件重新生成日志文件。

  例如，当-l参数指定为 gds.log，-R指定为20MB时，当gds.log达到20MB后就会新创建一个 "gds-2020-01-17_115425.log"文件。

  当-l指定的日志文件没有以log为后缀，例如：" gds.log.txt"， 则新创建的日志文件名为" gds.log-2020-01-19_122739.txt"。

  GDS启动时会检测-l参数设置的日志文件是否存在，如果存在则根据当前日期时间新生成一个日志文件，不会覆盖之前的日志文件。

- -H address_string

  设置允许哪些主机连接到GDS，参数需为CIDR格式，仅支持linux系统。需要配置多个不同网段时，使用“,”分隔。例如：-H 10.10.0.0/24,10.10.5.0/24。

- -e dir

  设置导入时产生的错误日志存放路径。

  默认值：数据文件目录。

- -E size

  设置导入产生的错误日志的上限值。

  取值范围：0<size<1TB，请使用正整数+单位的形式进行取值设置，单位支持KB、MB和GB。

- -S size

  设置导出单个文件大小上限。

  取值范围：1MB<size<100TB，请使用正整数+单位的形式进行取值设置，单位支持KB、MB和GB。如果使用KB，取值需要大于1024KB。

- -R size

  设置-l指定的GDS单个日志文件大小上限。

  取值范围：1MB<size<1TB，请使用正整数+单位的形式进行取值设置，单位支持KB、MB和GB。如果使用KB，取值需要大于1024KB。

  默认值：16MB。

- -t worker_num

  设置导入导出工作并发线程数目。

  取值范围：0<worker_num≤200，正整数。

  默认值：8。

  推荐值：普通文件导入导出场景取值：CPU核数*2；管道文件导入导出场景取值：64。

  说明

  当管道文件导入导出场景并发较大时，该值应不低于业务并发数。

- -s status_file

  设置状态文件，仅支持linux系统。

- -D

  后台运行GDS，仅支持linux系统。

- -r

  递归遍历目录（外表目录下的子目录）下文件，会递归读取location指定的目录层级下所有的同名文件，仅支持linux系统。

- --enable-ssl

  使用SSL认证的方式与集群通信。

- --ssl-dir cert_file

  在使用SSL认证方式时，指定认证证书的所在路径。

- --debug-level

  设置GDS端的debug日志级别，以控制GDS debug相关的日志输出。

  取值范围：0、1、2。

  - 0：仅打印导入导出相关的文件列表，日志量小，推荐在系统处于正常状态时使用设置。
  - 1：打印日志的完整信息，增加各节点的连接信息、session转换信息和一些数据统计。推荐仅在故障定位时开启。
  - 2：打印详细的交互日志以及所属状态，输出较大量的debug日志信息，以帮助故障定位分析。推荐仅在故障定位时开启。

- --pipe-timeout

  设置GDS操作管道文件的等待超时时间。

  取值范围：大于1s。请使用正整数+单位的形式进行取值设置，单位支持s、m和h。如：1小时可以设置为3600s、60m或者1h。

  默认值：1h/60m/3600s

  说明

  - 该参数的设置是为了避免人为或程序自身问题造成管道文件的一端长时间不读取或者不写入，导致管道另一端的读取或写入操作hang住。
  - 该参数表示的超时时间不是指GDS一个导入导出任务的最长时间，而是GDS对管道文件的每一次read/open/write的最大超时时间，当超过--pipe-timeout参数设置时间会向前端报错。

示例

启动一个GDS进程，其数据文件存放在“/data” 目录，IP为192.168.0.90，监听端口为5000。

```
gds_ctl start --host 192.168.0.90 -d /data/ -p 5000 -H 10.10.0.1/24 -D
```

启动一批GDS进程，其数据文件存放在“/data” 目录，IP为192.168.0.90、192.168.0.91、192.168.0.92，监听端口为5000。

```
gds_ctl start --host 192.168.0.90,192.168.0.91,192.168.0.92 -d /data/ -p 5000 -H 0/0 -D
```

批量关闭位于192.168.0.90、192.168.0.91、192.168.0.92节点上，端口是5000的GDS进程：

```
gds_ctl stop --host 192.168.0.90,192.168.0.91,192.168.0.92 -p 5000
```

批量重启位于192.168.0.90、192.168.0.91、192.168.0.92节点上，端口是5000的GDS进程：

```
gds_ctl restart  --host 192.168.0.90,192.168.0.91,192.168.0.92 -p 5000
```