GaussDB
2.x

Tool Reference for Primary+Standby Instances

Issue	01
Date	2024-06-06

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


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

Notice

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

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

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: https://www.huaweicloud.com/intl/en-us/

Contents

1 gsql

gsql, provided by GaussDB, is a database connection tool that runs in the command line. You can use **gsql** to connect to the server and perform operations and maintenance. In addition, **gsql** provides multiple **Advanced Features** for users.

1.1 Overview

Basic Features

- **Connect to the database**: By default, only the local server can be connected. To connect to a remote database, you must configure the server. For details, see "Database Quick Start > Connecting to a Database > Using gsql to Connect to a Database > Remotely Connecting to a Database" in the *Developer Guide*.

  NOTE

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

  To resolve this problem, see **Troubleshooting**.
- **Run SQL statements**: Interactively entered SQL statements and specified SQL statements in a file can be run.
- **Run meta-commands**: Meta-commands help the administrator view database object information, query cache information, format SQL output, and connect to a new database. For details about meta-commands, see **Meta-Command Reference**.

Advanced Features

Table 1-1 lists the advanced features of **gsql**.

Table 1-1 Advanced features of **gsql**

Feature Name	Description
Variable	**gsql** provides a variable feature that is similar to the **shell** command of Linux. The following **\set** meta-command of **gsql** can be used to set a variable: **\set** *varname value* To delete the variables set by the **\set** command, run the following command: **\unset** *varname* **NOTE** • A variable is a simple name-value pair. The value can be any characters in any length. • Variable names must consist of case-sensitive letters (including non-Latin letters), digits, and underscores (_). • If the **\set** *varname* meta-command (without the second parameter) is used, the variable is set without a value specified. • If the **\set** meta-command without parameters is used, values of all variables are displayed. For details about variable examples and descriptions, see **Variables**.
SQL substitution	Common SQL statements can be set to variables using the variable feature of **gsql** to simplify operations. For details about examples and descriptions about SQL substitution, see **SQL substitution**.
Customized prompt	Prompts of **gsql** can be customized. Prompts can be modified by changing the reserved three variables of **gsql**: *PROMPT1*, *PROMPT2*, and *PROMPT3*. These variables can be set to customized values or the values predefined by **gsql**. For details, see **Prompt**.
Historical client operation records	**gsql** can record historical client operations. This function is enabled by specifying the **-r** parameter when a client is connected. The number of historical records can be set using the **\set** command. For example, **\set HISTSIZE 50** indicates that the number of historical records is set to **50**. **\set HISTSIZE 0** indicates that the operation history is not recorded. **NOTE** • The default number of historical records is **32**. The maximum number of historical records is **500**. If interactively entered commands contain Chinese characters, only the UTF-8 encoding environment is supported. • For security reasons, the records containing sensitive words (such as PASSWORD, IDENTIFIED, GS_ENCRYPT_AES128, GS_DECRYPT_AES128, GS_ENCRYPT, GS_DECRYPT, PG_CREATE_PHYSICAL_REPLICATION_SLOT_EXTERN, SECRET_ACCESS_KEY, SECRETKEY, and CREATE_CREDENTIAL) are regarded sensitive and not recorded in historical information. This indicates that you cannot view these records in command output histories.

- Variable

  To set a variable, run the **\set** meta-command of **gsql**. For example, to set variable *foo* to **bar**, run the following command:

  ```
  openGauss=# \set foo bar
  ```

  To reference the value of a variable, add a colon (:) before the variable. For example, to view the value of variable *foo*, run the following command:

  ```
  openGauss=# \echo :foo
  bar
  ```

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

  gsql pre-defines some special variables and plans the values of these variables. To ensure compatibility with later versions, do not use these variables for other purposes. For details about special variables, see **Table 1-2**.

  NOTE

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

  Table 1-2 Settings of special variables

Variable	Setting Method	Description
DBNAME	\set DBNAME *dbname*	Name of the connected database. This variable is set again when a database is connected.
ECHO	\set ECHO all | queries	• If this variable is set to **all**, only the query information is displayed. This has the same effect as specifying the **-a** parameter when **gsql** is used to connect to a database. • If this variable is set to **queries**, the command line and query information are displayed. This has the same effect as specifying the **-e** parameter when **gsql** is used to connect to a database.

Variable	Setting Method	Description
ECHO_HIDDEN	\set ECHO_HIDDEN on | off | noexec	When a meta-command (such as **\dg**) is used to query database information, the value of this variable determines the query behavior. ● If this variable is set to **on**, the query statements that are called by the meta-command are displayed, and then the query result is displayed. This has the same effect as specifying the **-E** parameter when **gsql** is used to connect to a database. ● If this variable is set to **off**, only the query result is displayed. ● If this variable is set to **noexec**, only the query information is displayed, and the query is not run.
ENCODING	\set ENCODING *encoding*	Character set encoding of the current client.
FETCH_COUNT	\set FETCH_COUNT *variable*	● If the value is an integer greater than **0**, for example, *n*, *n* lines will be selected from the result set to the cache and displayed on the screen when the **SELECT** statement is run. ● If this variable is not set or set to a value less than or equal to **0**, all results are selected at a time to the cache when the **SELECT** statement is run. **NOTE** A proper variable value helps reduce the memory usage. The recommended value range is from 100 to 1000.
HISTCONTROL	\set HISTCONTROL ignorespace | ignoredups | ignoreboth | none	● **ignorespace**: A line started with a space is not written to the historical record. ● **ignoredups**: A line that exists in the historical record is not written to the historical record. ● **ignoreboth**, **none**, or other values: All the lines read in interaction mode are saved in the historical record. **NOTE** **none** indicates that **HISTCONTROL** is not set.
HISTFILE	\set HISTFILE *filename*	Specifies the file for storing historical records. The default value is **~/.bash_history**.

Variable	Setting Method	Description
HISTSIZE	\set HISTSIZE *size*	Specifies the number of commands to store in the command history. The default value is **500**.
HOST	\set HOST *hostname*	Specifies the name of a connected host.
IGNOREEOF	\set IGNOREEOF *variable*	• If this variable is set to a number, for example, **10**, the first nine EOF characters (generally **Ctrl**+**C**) entered in **gsql** are neglected and the **gsql** program exits when the tenth **Ctrl**+**C** is entered. • If this variable is set to a non-numeric value, the default value is **10**. • If this variable is deleted, **gsql** exits when an EOF is entered.
LASTOID	\set LASTOID *oid*	Specifies the last OID, which is the value returned by an **INSERT** or **lo_import** command. This variable is valid only before the output of the next SQL statement is displayed.
ON_ERROR_ROLLBACK	\set ON_ERROR_ROLLBACK on | interactive | off	• If the value is **on**, an error that may occur in a statement in a transaction block is ignored and the transaction continues. • If the value is **interactive**, the error is ignored only in an interactive session. • If the value is **off** (default value), the error triggers the rollback of the transaction block. In **on_error_rollback-on** mode, a **SAVEPOINT** is set before each statement of a transaction block, and an error triggers the rollback of the transaction block.
ON_ERROR_STOP	\set ON_ERROR_STOP on | off	• **on**: specifies that the execution stops if an error occurs. In interactive mode, **gsql** returns the output of executed commands immediately. • **off** (default value): specifies that an error, if occurring during the execution, is ignored, and the execution continues.
PORT	\set PORT *port*	Specifies the port number of a connected database.

Variable	Setting Method	Description
USER	\set USER *username*	Specifies the database user you are currently connected as.
VERBOSITY	\set VERBOSITY terse | default | verbose	This variable can be set to **terse**, **default**, or **verbose** to control redundant lines of error reports. • **terse**: Only critical and major error texts and text locations are returned (which is generally suitable for single-line error information). • **default**: Critical and major error texts and text locations, error details, and error messages (possibly involving multiple lines) are all returned. • **verbose**: All error information is returned.

- SQL substitution

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

```
openGauss=# \set foo 'HR.areaS'
openGauss=# select * from :foo;
 area_id |       area_name
---------+------------------------
       4 | Middle East and Africa
       3 | Asia
       1 | Europe
       2 | Americas
(4 rows)
```

  The above command queries the HR.areaS table.

  NOTICE

  The value of the variable is copied literally, so it can even contain unbalanced quotation marks or backslash commands. Therefore, the input content must be meaningful.

- Prompt

  The **gsql** prompt can be set using the three variables in **Table 1-3**. These variables consist of characters and special escape characters.

Table 1-3 Prompt variables

Variable	Description	Example

PROMPT1	Specifies the normal prompt used when **gsql** requests a new command.
	The default value of *PROMPT1* is:
	`%/%R%#`
		PROMPT1 can be used to change the prompt.
		• Change the prompt to **[local]**:
		`openGauss=> \set PROMPT1 %M`
		`[local:/tmp/gaussdba_mppdb]`
		• Change the prompt to **name**:
		`openGauss=> \set PROMPT1 name`
		`name`
		• Change the prompt to **=**:
		`openGauss=> \set PROMPT1 %R`
		`=`

PROMPT2	Specifies the prompt displayed when more input is expected because the command that is not terminated with a semicolon (;) or a quote (') is not closed.	*PROMPT2* can be used to display the prompt.
		`openGauss=# \set PROMPT2 TEST`
		`openGauss=# select * from HR.areaS TEST;`
		` area_id |  area_name`
		`---------+--------------------`
		`  1 | Europe`
		`  2 | Americas`
		`  4 | Middle East and Africa`
		`  3 | Asia`
		`(4 rows))`

PROMPT3	Specifies the prompt displayed when the **COPY** statement (such as **COPY FROM STDIN**) is run and data input is expected.	*PROMPT3* can be used to display the **COPY** prompt.
		`openGauss=# \set PROMPT3 '>>>>'`
		`openGauss=# copy HR.areaS from STDIN;`
		`Enter data to be copied followed by a newline.`
		`End with a backslash and a period on a line by itself.`
		`>>>>1 aa`
		`>>>>2 bb`
		`>>>>\.`

The value of the selected prompt variable is printed literally. However, a value containing a percent sign (%) is replaced by the predefined contents depending on the character following the percent sign (%). For details about the defined substitutions, see **Table 1-4**.

Table 1-4 Defined substitutions

Symbol	Description
%M	Replaced with the full host name (with domain name). The full name is **[local]** if the connection is over a Unix domain socket, or **[local:/dir/name]** if the Unix domain socket is not at the compiled default location.
%m	Replaced with the host name truncated at the first dot. It is **[local]** if the connection is over a Unix domain socket.

Symbol	Description
%>	Replaced with the number of the port that the host is listening on.
%n	Replaced with the database session username.
%/	Replaced with the name of the current database.
%~	Similar to **%/**. However, the output is tilde (~) if the database is your default database.
%#	Uses **#** if the session user is the database administrator. Otherwise, uses **>**.
%R	• In *PROMPT1* normally **=**, but **^** if in single-line mode, or **!** if the session is disconnected from the database (which can happen if **\connect** fails). • In *PROMPT2* %R is replaced with a hyphen (-), an asterisk (*), a single or double quotation mark, or a dollar sign ($), depending on whether **gsql** expects more input because the query is inside a /*...*/ comment or inside a quoted or dollar-escaped string.
%x	Replaced with the transaction status. • An empty string when it is not in a transaction block • An asterisk (*) when it is in a transaction block • An exclamation mark (!) when it is in a failed transaction block • A question mark (?) when the transaction status is indefinite (for example, because there is no connection).
%digits	Replaced with the character with the specified byte.
%:name	Replaced with the value of the *name* variable of **gsql**.
%command	Replaced with the command output, similar to substitution with the "^" symbol.
%[. . . %]	Prompts may contain terminal control characters which, for example, change the color, background, or style of the prompt text, or change the title of the terminal window. For example: openGauss=> \set PROMPT1 '%[%033[1;33;40m%]%n@%/%R %[%033[0m%]%#' The output is a boldfaced (1;) yellow-on-black (33;40) prompt on VT100-compatible, color-capable terminals.

Environment Variables

Table 1-5 Environment variables related to **gsql**

Name	Description
COLUMNS	If **\set columns** is set to **0**, this parameter controls the width of the wrapped format. This width determines whether to change the wide output mode into the vertical output mode if automatic expansion is enabled.
PAGER	If the query results do not fit on the screen, they are redirected through this command. You can use the **\pset** command to disable the pager. Typically, the **more** or **less** command is used for viewing the query result page by page. The default is platform-dependent. **NOTE** Display of the **less** command is affected by the *LC_CTYPE* environment variable.
PSQL_EDITOR	The **\e** and **\ef** commands use the editor specified by the environment variables. The variables are examined in the order listed. The default editor on Unix is vi.
EDITOR	
VISUAL	

PSQL_EDITOR_LINENUMBER_ARG	When the **\e** or **\ef** command is used with a line number parameter, this variable specifies the command-line parameter used to pass the starting line number to the editor. For editors, such as Emacs or vi, this is a plus sign. Include a space in the value of the variable if space is needed between the option name and the line number. For example:
	`PSQL_EDITOR_LINENUMBER_ARG = '+'`
	`PSQL_EDITOR_LINENUMBER_ARG='--line '`
	A plus sign (+) is used by default on Unix.

PSQLRC	Specifies the location of the user's .gsqlrc file.
SHELL	Has the same effect as the **\!** command.
TMPDIR	Specifies the directory for storing temporary files. The default value is **/tmp**.

1.2 Usage Guidelines

Prerequisites

The user using **gsql** must have the permission to access the database.

Background

You can use the **gsql** command to connect to the local database or remote database. When connecting to the remote database, enable remote connection on

the server. For details, see "Database Quick Start > Connecting to a Database > Using gsql to Connect to a Database > Remotely Connecting to a Database" in the *Developer Guide*.

Procedure

Step 1 Connect to the GaussDB server using the **gsql** tool.

The **gsql** tool uses the **-d** parameter to specify the target database name, the **-U** parameter to specify the database username, the **-h** parameter to specify the host name, and the **-p** parameter to specify the port number.

NOTE

If the database name is not specified, the default database name generated during initialization will be used. If the database username is not specified, the current OS username will be used by default. If a variable does not belong to any parameter (such as **-d** and **-U**), and **-d** is not specified, the variable will be used as the database name. If **-d** is specified but **-U** is not specified, the variable will be used as the database username.

Example 1: Connect to the 8000 port of the local **gaussdb** database as user **omm**.

```
gsql -d gaussdb -p 8000
```

Example 2: Connect to the 8000 port of the remote **gaussdb** database as user **jack**.

```
gsql -h 10.180.123.163 -d gaussdb -U jack -p 8000
```

In a centralized database instance, when connecting to the primary DN, you can use commas (,) to separate the IP addresses of DNs and add them to the end of **-h**. **gsql** connects to each IP address in sequence to check whether the current DN is the primary DN. If no, **gsql** disconnects from the current IP address and attempts to connect to the next IP address until the primary DN is found.

```
gsql -h 10.180.123.163,10.180.123.164,10.180.123.165 -d gaussdb -U jack -p 8000
```

Example 3: If **gaussdb** and **omm** are not parameter values, they are interpreted as the database name and the username, respectively.

```
gsql gaussdb omm -p 8000
```

Equals

```
gsql -d gaussdb -U omm -p 8000
```

For details about the **gsql** parameters, see **Command Reference**.

Step 2 Run a SQL statement.

The following takes creating database **human_staff** as an example:

```
CREATE DATABASE human_staff;
CREATE DATABASE
```

Ordinarily, input lines end when a command-terminating semicolon is reached. If the command is sent and executed without any error, the command output is displayed on the screen.

Step 3 Execute gsql meta-commands.

The following takes all GaussDB databases and description information as an example:


```
openGauss=# \l
                                  List of databases
     Name      |  Owner   | Encoding  | Collate | Ctype |   Access privileges
---------------+----------+-----------+---------+-------+-----------------------
 human_resource | omm | SQL_ASCII | C       | C     |
 postgres       | omm | SQL_ASCII | C       | C     |
 template0      | omm | SQL_ASCII | C       | C     | =c/omm          +
                |     |           |         |       | omm=CTc/omm
 template1      | omm | SQL_ASCII | C       | C     | =c/omm          +
                |     |           |         |       | omm=CTc/omm
 human_staff    | omm | SQL_ASCII | C       | C     |
(5 rows)
```

For details about **gsql** meta-commands, see **Meta-Command Reference**.

----End

Example

The example shows how to spread a command over several lines of input. Note the prompt change:

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

Query the table definition:

```
openGauss=# \d HR.areaS
              Table "hr.areas"
  Column   |         Type          | Modifiers
-----------+-----------------------+-----------
 area_id   | numeric               | not null
 area_name | character varying(25) |
```

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

```
openGauss=# INSERT INTO HR.areaS (area_ID, area_NAME) VALUES (1, 'Europe');
INSERT 0 1
openGauss=# INSERT INTO HR.areaS (area_ID, area_NAME) VALUES (2, 'Americas');
INSERT 0 1
openGauss=# INSERT INTO HR.areaS (area_ID, area_NAME) VALUES (3, 'Asia');
INSERT 0 1
openGauss=# INSERT INTO HR.areaS (area_ID, area_NAME) VALUES (4, 'Middle East and Africa');
INSERT 0 1
```

Change the prompt.

```
openGauss=# \set PROMPT1 '%n@%m %~%R%#'
omm@[local] openGauss=#
```

Query the table:

```
omm@[local] openGauss=# SELECT * FROM HR.areaS;
 area_id |       area_name
---------+------------------------
       1 | Europe
       4 | Middle East and Africa
       2 | Americas
       3 | Asia
(4 rows)
```

Use the **\pset** command to display the table in different ways:

```
omm@[local] openGauss=# \pset border 2
Border style is 2.
```



```
omm@[local] openGauss=# SELECT * FROM HR.areaS;
+---------+------------------------+
| area_id |       area_name        |
+---------+------------------------+
|       1 | Europe                 |
|       2 | Americas               |
|       3 | Asia                   |
|       4 | Middle East and Africa |
+---------+------------------------+
(4 rows)
omm@[local] openGauss=# \pset border 0
Border style is 0.
omm@[local] openGauss=# SELECT * FROM HR.areaS;
area_id       area_name
------- ----------------------
      1 Europe
      2 Americas
      3 Asia
      4 Middle East and Africa
(4 rows)
```

Use the meta-command:

```
omm@[local] openGauss=# \a \t \x
Output format is unaligned.
Showing only tuples.
Expanded display is on.
omm@[local] openGauss=# SELECT * FROM HR.areaS;
area_id|2
area_name|Americas

area_id|1
area_name|Europe

area_id|4
area_name|Middle East and Africa

area_id|3
area_name|Asia
omm@[local] openGauss=#
```

1.3 Obtaining Help Information

Procedure

- When connecting to the database, run the following command to obtain the help information:

  ```
  gsql --help
  ```

  The following help information is displayed:

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

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

- When connecting to the database, run the following command to obtain the help information:

  ```
  help
  ```

  The following help information is displayed:

  ```
  You are using gsql, the command-line interface to gaussdb.
  Type:  \copyright for distribution terms
  ```

```
\h for help with SQL commands
\? for help with gsql commands
\g or terminate with semicolon to execute query
\q to quit
```

Examples

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

```
gsql -d gaussdb -p 8000
```

gaussdb is the name of the database to be connected, and 8000 is the port number of the database primary node.

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

```
gsql ((GaussDB Kernel VxxxRxxxCxx build 290d125f) compiled at 2020-05-08 02:59:43 commit 2143 last mr 131)
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.
```

Step 2 View the **gsql** help information. For details, see **Table 1-6**.

Table 1-6 gsql online help

Description	Example
Query the copyright.	\copyright

View help information about SQL statements supported by GaussDB.	View help information about SQL statements supported by GaussDB.
	For example, view all SQL statements supported by GaussDB.
	`openGauss=# \h`
	`Available help:`
	`  ABORT`
	`  ALTER AGGREGATE`
	`  ALTER APP WORKLOAD GROUP`
	`... ...`
	For example, view parameters of the **CREATE DATABASE** command:
	`openGauss=# \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 ]}[...] ];`

Description	Example
View the help information about **gsql** commands.	For example, view commands supported by **gsql**. openGauss=# \? General \copyright show openGauss 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 ... ...

----End

1.4 Command Reference

For details about gsql parameters, see **Table 1-7**, **Table 1-8**, **Table 1-9**, and **Table 1-10**.

Table 1-7 Common parameters

Parameter	Description	Value Range
-c, --command=COMMAND	Specifies that gsql is to run a string command and then exit.	-
-d, --dbname=DBNAME	Specifies the name of the database to connect to. In addition, gsql allows you to use extended database names, that is, connection strings in the format of **'postgres[ql]://[user[:password]@][netloc][:port][, ...][/dbname][?param1=value1&...]'** or **'[key=value][...]'** as database names. gsql parses connection information from the connection strings and preferentially uses the information.	String
-f, --file=FILENAME	Specifies that files are used as the command source instead of interactively-entered commands. After the files are processed, gsql exits. If *FILENAME* is a hyphen (-), then standard input is read.	An absolute path or relative path that meets the OS path naming convention
-l, --list	Lists all available databases and then exits.	-
-v, --set, --variable=NAME=VALUE	Sets gsql variable *NAME* to **VALUE**. For details about variable examples and descriptions, see **Variables**.	-

Parameter	Description	Value Range
-X, --no-gsqlrc	Does not read the startup file (neither the system-wide **gsqlrc** file nor the user's **~/.gsqlrc** file). **NOTE** The startup file is **~/.gsqlrc** by default or it can be specified by the environment variable *PSQLRC*.	-
-1 ("one"), --single-transaction	When gsql uses the **-f** parameter to execute a script, START TRANSACTION/ COMMIT are added to the start and end of the script, respectively, so that the script is executed as one transaction. This ensures that the script is executed successfully. If the script cannot be executed, the script is invalid. **NOTE** If the script has used **START TRANSACTION**, **COMMIT**, or **ROLLBACK**, this parameter is invalid.	-
-?, --help	Displays help information about gsql command parameters, and exits.	-
-V, --version	Prints the gsql version information and exits.	-

Table 1-8 Input and output parameters

Parameter	Description	Value Range
-a, --echo-all	Prints all input lines to standard output as they are read. **CAUTION** When this parameter is used in some SQL statements, the sensitive information, such as user password, may be disclosed. Use this parameter with caution.	-
-e, --echo-queries	Displays all queries sent to the server to the standard output as well. **CAUTION** When this parameter is used in some SQL statements, the sensitive information, such as user password, may be disclosed. Use this parameter with caution.	-
-E, --echo-hidden	Echoes the actual queries generated by **\d** and other backslash commands.	-

Parameter	Description	Value Range
-k, --with-key=KEY	Uses gsql to decrypt imported encrypted files. **NOTICE** • For key characters, such as the single quotation mark (') or double quotation mark (") in shell commands, Linux shell checks whether the input single quotation mark (') or double quotation mark (") matches. If no match is found, Linux shell does not enter the gsql program until input is complete. • Stored procedures and functions cannot be decrypted and imported.	-
-L, --log-file=FILENAME	Writes normal output source and all query output into the **FILENAME** file. **CAUTION** • When this parameter is used in some SQL statements, the sensitive information, such as user password, may be disclosed. Use this parameter with caution. • This parameter retains only the query result in the corresponding file, so that the result can be easily found and parsed by other invokers (for example, automatic O&M scripts). Logs about gsql operations are not retained.	An absolute path or relative path that meets the OS path naming convention
-m, --maintenance	Allows connections to the database during two-phase transaction recovery. **NOTE** The parameter is for engineers only. When this parameter is used, gsql can be connected to the standby server to check data consistency between the primary and standby server.	-
-n, --no-libedit	Closes command line editing.	-
-o, --output=FILENAME	Puts all query output into the **FILENAME** file.	An absolute path or relative path that meets the OS path naming convention
-q, --quiet	Indicates the quiet mode and no additional information will be printed.	By default, gsql displays various information.

Parameter	Description	Value Range
-s, --single-step	Runs in single-step mode. It indicates that the user is prompted before each command is sent to the server. This option can be also used for canceling execution. Use this option to debug scripts. **CAUTION** When this parameter is used in some SQL statements, the sensitive information, such as user password, may be disclosed. Use this parameter with caution.	-
-S, --single-line	Runs in single-line mode where a line break terminates a command, as a semicolon does.	-
-C, --enable-client-encryption	When **-C** is used to connect to a local or remote database, you can use this option to enable the encrypted database function.	-

Table 1-9 Output format parameters

Parameter	Description	Value Range
-A, --no-align	Switches to unaligned output mode.	The default output mode is aligned.
-F, --field-separator=STRING	Specifies the field separator. The default is the vertical bar (|).	-
-H, --html	Turns on the HTML tabular output.	-
-P, --pset=VAR[=ARG]	Specifies the print option in the **\pset** format in the command line. **NOTE** The equal sign (=), instead of the space, is used here to separate the name and value. For example, enter **-P format=latex** to set the output format to **LaTeX**.	-
-R, --record-separator=STRING	Sets the record separator.	-
-r	Enables the editing mode on the client.	This function is disabled by default.

Parameter	Description	Value Range
-t, --tuples-only	Prints only tuples.	-
-T, --table-attr=TEXT	Specifies options to be placed within the HTML table tag. Use this parameter with the **-H,--html** parameter to specify the output to the HTML format.	-
-x, --expanded	Turns on the expanded table formatting mode.	-
-z, --field-separator-zero	Sets the field separator in the unaligned output mode to be blank. Use this parameter with the **-A, --no-align** parameter to switch to unaligned output mode.	-
-0, --record-separator-zero	Sets the record separator in the unaligned output mode to be blank. Use this parameter with the **-A, --no-align** parameter to switch to unaligned output mode.	-
-2, --pipeline	Uses a pipe to transmit the password. This parameter cannot be used on devices and must be used together with the **-c** or **-f** parameter.	-

Table 1-10 Connection parameters

Parameter	Description	Value Range
-h, --host=HOSTNAME	Specifies the host name of the machine on which the server is running or the directory for the Unix-domain socket.	If the host name is omitted, gsql connects to the server of the local host over the Unix domain socket or over TCP/IP to connect to local host without the Unix domain socket.

Parameter	Description	Value Range
-p, --port=PORT	Specifies the port number of the database server. You can modify the default port number using the **-p, --port=PORT** parameter.	The default port number can be specified by using compilation parameters. If the port number is not specified, the default port number **5432** is used.
-U, --username=USERNAME	Specifies the user that connects to the database. **NOTE** • If this parameter is specified, you also need to enter your password for identity authentication when connecting to the database. You can enter the password interactively or use the **-W** parameter to specify a password. • To connect to a database, add an escape character before any dollar sign ($) in the username.	String. The default user is the current user that operates the system.
-W, --password=PASSWORD	Specifies a password when the **-U** parameter is used to connect to the local database or a remote database. **NOTE** • When the server where the primary database node is located connects to the local primary database node, the trust connection is used by default and this parameter is ignored. • To connect to a database, add an escape character before any backslash (\) or back quote (`) in the password. • If this parameter is not specified but database connection requires your password, you will be prompted to enter your password in interactive mode. The maximum length of the password is 999 bytes, which is restricted by the maximum value of the GUC parameter **password_max_length**.	String

1.5 Meta-Command Reference

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

Precautions

- The format of the **gsql** meta-command is a backslash (\) followed by a command verb, and then a parameter. The parameters are separated from the command verb and from each other by any number of whitespace characters.
- To include whitespace characters into an argument, you must quote them with a single straight quotation mark. To include a single straight quotation mark into such an argument, precede it by a backslash. Anything contained in single quotation marks is furthermore subject to C-like substitutions for \n (new line), \t (tab), \b (backspace), \r (carriage return), \f (form feed), \digits (octal), and \xdigits (hexadecimal).
- Within a parameter, text enclosed in double quotation marks ("") is taken as a command line input to the shell. The output of the command (with any trailing newline removed) is taken as the argument value.
- If an unquoted argument begins with a colon (:), the argument is taken as a **gsql** variable and the value of the variable is used as the argument value instead.
- Some commands take an SQL identifier (such as a table name) as a parameter. These parameters follow the SQL syntax rules: Unquoted letters are forced to lowercase, while double quotation marks ("") protect letters from case conversion and allow incorporation of whitespace into the identifier. Within double quotation marks, paired double quotation marks reduce to a single double quotation mark in the result name. For example, **FOO"BAR"BAZ** is interpreted as **fooBARbaz**, and **"Aweird""name"** becomes **A weird"name**.
- Parsing for arguments stops when another unquoted backslash is found. This is taken as the beginning of a new meta-command. The special sequence \\ (two backslashes) marks the end of parameters and continues parsing SQL statements if any. In this way, SQL and **gsql** commands can be freely mixed in a line. But in any case, the arguments of a meta-command cannot continue beyond the end of the line.

Meta-command

For details about meta-commands, see **Table 1-11**, **Table 1-12**, **Table 1-13**, **Table 1-14**, **Table 1-16**, **Table 1-18**, **Table 1-19**, **Table 1-20**, and **Table 1-22**.

NOTICE

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

Table 1-11 Common meta-commands

Parameter	Description	Value Range
\copyright	Displays GaussDB version and copyright information.	-

Parameter	Description	Value Range
\g [FILE] or ;	Performs a query operation and sends the result to a file or pipe.	-
\h(\help) [NAME]	Provides syntax help on the specified SQL statement.	If the name is not specified, then **gsql** will list all the commands for which syntax help is available. If the name is an asterisk (*), syntax help on all SQL statements is displayed.

Parameter	Description	Value Range
\parallel [on [num]|off]	Controls the parallel execution function. • **on**: The switch is enabled and the maximum number of concurrently executed tasks is **num**. • **off**: This switch is disabled. **NOTE** • Parallel execution is not allowed in a running transaction and a transaction is not allowed to be started during parallel execution. • Parallel execution of **\d** meta-commands is not allowed. • If **SELECT** statements are run concurrently, customers can accept the problem that the return results are displayed randomly but they cannot accept it if a core dump or process response failure occurs. • **SET** statements are not allowed in concurrent tasks because they may cause unexpected results. • Temporary tables cannot be created in parallel. If temporary tables are required, create them before parallel execution is enabled, and use them only in the parallel execution. Temporary tables cannot be created in parallel execution. • When **\parallel** is executed, *num* independent gsql processes can be connected to the database server. • The total duration of all **\parallel** tasks cannot exceed **session_timeout**. Otherwise, the connection may fail during concurrent execution. • One or more commands following **\parallel on** will be executed only after **\parallel off** is executed. Therefore, **\parallel on** must be followed by **\parallel off**. Otherwise, the commands following **\parallel on** cannot be executed.	The default value of *num* is **1024**. **NOTICE** • The maximum number of connections allowed by the server is determined based on **max_connection** and the number of current connections. • Set the value of *num* based on the allowed number of connections.
\q	Exits the gsql program. This command is executed only when a script terminates in a script file.	-

Table 1-12 Query buffer meta-commands

Parameter	Description
\e [FILE] [LINE]	Uses an external editor to edit the query buffer or file.
\ef [FUNCNAME [LINE]]	Edits the function definition using an external editor. If **LINE** is specified, the cursor will point to the specified line of the function body.
\p	Prints the current query buffer to the standard output.
\r	Resets or clears the query buffer.
\w FILE	Outputs the current query buffer to a file.

Table 1-13 Input/Output commands

Parameter	Description
\copy { table [(column_list)] | (query) } { from | to } { filename | stdin | stdout | pstdin | pstdout } [with] [binary] [oids] [delimiter [as] 'character'] [useeof] [null [as] 'string'] [csv [header] [quote [as] 'character'] [escape [as] 'character'] [force quote column_list | *] [force not null column_list]] [parallel integer]	After logging in to the database on any **gsql** client, you can import and export data. This is an operation of running the **SQL COPY** command, but not the server that reads or writes data to a specified file. Instead, data is transferred between the server and the local file system. In this way, local user permissions instead of server permissions are required for file access, and the user permissions do not need to be initialized. **NOTE** **\COPY** is applicable only to small-batch data import in a good format. It does not preprocess invalid characters and does not have the error tolerance capability. GDS or **COPY** is preferred for data import. \COPY specifies the number of clients to import data to implement parallel import of data files. Currently, the value ranges from 1 to 8. The parallel import using \COPY has the following constraints: Parallel import of temporary tables is not supported. Parallel import within transactions is not supported. Parallel import of binary files is not supported. Parallel import of data encrypted using AES-128 is not supported. The COPY option contains EOL. In these cases, even if the parallel parameter is specified, a non-parallel process is performed.
\echo [STRING]	Writes character strings to the standard output.

Parameter	Description
\i FILE	Reads content from *FILE* and uses them as the input for a query.
\i+ FILE KEY	Runs commands in an encrypted file.
\ir FILE	Similar to **\i**, but resolves relative path names differently.
\ir+ FILE KEY	Similar to **\i+**, but resolves relative path names differently.
\o [FILE]	Saves all query results to a file.
\qecho [STRING]	Writes character strings to the query output flow.

NOTE

In Table 1-14, **S** indicates displaying the system object and **+** indicates displaying the additional description information of the object. **PATTERN** specifies the name of an object to be displayed.

Table 1-14 Information display meta-commands

Parameter	Description	Value Range	Example
\d[S+]	Lists all tables, views, and sequences of all schemas in the search_path. When objects with the same name exist in different schemas in **search_path**, only the object in the schema that ranks first in **search_path** is displayed.	-	Lists all tables, views, and sequences of all schemas in the search_path. `openGauss=# \d`
\d[S+] NAME	Lists the structure of specified tables, views, and indexes.	-	Lists the structure of table **a**. `openGauss=# \dtable+ a`
\d+ [PATTERN]	Lists all tables, views, and indexes.	If **PATTERN** is specified, only tables, views, and indexes whose names match **PATTERN** are displayed.	Lists all tables, views, and indexes whose names start with **f**. `openGauss=# \d+ f*`

Parameter	Description	Value Range	Example
\da[S] [PATTERN]	Lists all available aggregate functions, together with the data type they perform operations on and the return value types.	If **PATTERN** is specified, only aggregate functions whose names match **PATTERN** are displayed.	Lists all available aggregate functions whose names start with **f**, together with their return value types and the data types. `openGauss=# \da f*`
\db[+] [PATTERN]	Lists all available tablespaces.	If **PATTERN** is specified, only tablespaces whose names match **PATTERN** are displayed.	Lists all available tablespaces whose names start with **p**. `openGauss=# \db p*`
\dc[S+] [PATTERN]	Lists all available conversions between character-set encodings.	If **PATTERN** is specified, only conversions whose names match **PATTERN** are displayed.	Lists all available conversions between character-set encodings. `openGauss=# \dc *`
\dC[+] [PATTERN]	Lists all available type conversions. PATTERN must be the actual type name and cannot be an alias.	If **PATTERN** is specified, only conversions whose names match **PATTERN** are displayed.	Lists all type conversions whose patten names start with **c**. `openGauss=# \dC c*`
\dd[S] [PATTERN]	Lists descriptions about objects matching **PATTERN**.	If no parameter is specified, all visible objects are displayed. The objects include aggregations, functions, operators, types, relations (tables, views, indexes, sequences, and large objects), and rules.	Lists all visible objects. `openGauss=# \dd`
\ddp [PATTERN]	Lists all default permissions.	If **PATTERN** is specified, only permissions whose names match **PATTERN** are displayed.	Lists all default permissions. `openGauss=# \ddp`

Parameter	Description	Value Range	Example
\dD[S+] [PATTERN]	Lists all available domains.	If **PATTERN** is specified, only domains whose names match **PATTERN** are displayed.	Lists all available domains. openGauss=# \dD
\ded[+] [PATTERN]	Lists all Data Source objects.	If **PATTERN** is specified, only objects whose names match **PATTERN** are displayed.	Lists all Data Source objects. openGauss=# \ded
\det[+] [PATTERN]	Lists all external tables.	If **PATTERN** is specified, only tables whose names match **PATTERN** are displayed.	Lists all external tables. openGauss=# \det
\des[+] [PATTERN]	Lists all external servers.	If **PATTERN** is specified, only servers whose names match **PATTERN** are displayed.	Lists all external servers. openGauss=# \des
\deu[+] [PATTERN]	Lists all user mappings.	If **PATTERN** is specified, only information whose name matches **PATTERN** is displayed.	Lists all user mappings. openGauss=#\deu
\dew[+] [PATTERN]	Lists all encapsulated external data.	If **PATTERN** is specified, only data whose name matches **PATTERN** is displayed.	Lists all encapsulated external data. openGauss=# \dew

Parameter	Description	Value Range	Example
\df[antw][S+] [PATTERN]	Lists all available functions, together with their parameters and return types. **a** indicates an aggregate function, **n** indicates a common function, **t** indicates a trigger, and **w** indicates a window function.	If **PATTERN** is specified, only functions whose names match **PATTERN** are displayed.	Lists all available functions, together with their parameters and return types. openGauss=# \df
\dF[+] [PATTERN]	Lists all text search configuration information.	If **PATTERN** is specified, only configurations whose names match **PATTERN** are displayed.	Lists all text search configuration information. openGauss=# \dF+
\dFd[+] [PATTERN]	Lists all text search dictionaries.	If **PATTERN** is specified, only dictionaries whose names match **PATTERN** are displayed.	Lists all text search dictionaries. openGauss=# \dFd
\dFp[+] [PATTERN]	Lists all text search analyzers.	If **PATTERN** is specified, only analyzers whose names match **PATTERN** are displayed.	Lists all text search analyzers. openGauss=# \dFp
\dFt[+] [PATTERN]	Lists all text search templates.	If **PATTERN** is specified, only templates whose names match **PATTERN** are displayed.	Lists all text search templates. openGauss=# \dFt
\dg[+] [PATTERN]	Lists all database roles. **NOTE** Since the concepts of "users" and "groups" have been unified into "roles", this command is now equivalent to **\du**. Both commands are retained to ensure compatibility with earlier versions.	If **PATTERN** is specified, only roles whose names match **PATTERN** are displayed.	Lists all database roles named **j?e** (the question mark (?) indicates any character). openGauss=# \dg j?e
\dl	This is an alias for **\lo_list**, which shows a list of large objects.	-	Lists all large objects. openGauss=# \dl

Parameter	Description	Value Range	Example
\dL[S+] [PATTERN]	Lists all available program languages.	If **PATTERN** is specified, only languages whose names match **PATTERN** are displayed.	Lists all available program languages. openGauss=# \dL
\dm[S+] [PATTERN]	Lists materialized views.	If **PATTERN** is specified, only materialized views whose names match **PATTERN** are displayed.	Lists materialized views. openGauss=# \dm
\dn[S+] [PATTERN]	Lists all schemas (namespace). If **+** is added to the command, the permission and description of each schema are listed.	If **PATTERN** is specified, only schemas whose names match **PATTERN** are displayed. By default, only schemas you created are displayed.	Lists information about all schemas whose names start with **d**. openGauss=# \dn+ d*
\do[S] [PATTERN]	Lists available operators with their operand and return types.	If **PATTERN** is specified, only operators whose names match **PATTERN** are displayed. By default, only the operators created by the user are listed.	Lists available operators with their operand and return types. openGauss=# \do
\dO[S+] [PATTERN]	Lists collation rules.	If **PATTERN** is specified, only rules whose names match **PATTERN** are displayed. By default, only user-created rules are shown.	Lists collation rules. openGauss=# \dO

Parameter	Description	Value Range	Example

\dp [PATTERN]	Lists tables, views, and related permissions.
	The following result about **\dp** is displayed:
	`rolename=xxxx/yyyy --Assigns permissions to a role.`
	`=xxxx/yyyy --Assigns permissions to public.`
	xxxx indicates the assigned permissions, and *yyyy* indicates the roles with the assigned permissions. For details about permission descriptions, see **Table 1-15**.
		If **PATTERN** is specified, only tables and views whose names match **PATTERN** are displayed.	Lists tables, views, and related permissions.
			`openGauss=# \dp`

\drds [PATTERN1 [PATTERN2]]	Lists all parameters that have been modified. These settings can be for roles, for databases, or for both. **PATTERN1** and **PATTERN2** indicate a role pattern and a database pattern, respectively.	If **PATTERN** is specified, only collations rules whose names match **PATTERN** are displayed. If the default value is used or ***** is specified, all settings are listed.	Lists all the modified configuration parameters of the database. `openGauss=# \drds * dbname`
\dT[S+] [PATTERN]	Lists all data types.	If **PATTERN** is specified, only types whose names match **PATTERN** are displayed.	Lists all data types. `openGauss=# \dT`
\du[+] [PATTERN]	Lists all database roles. **NOTE** Since the concepts of "users" and "groups" have been unified into "roles", this command is now equivalent to **\dg**. Both commands are retained to ensure compatibility with earlier versions.	If **PATTERN** is specified, only roles whose names match **PATTERN** are displayed.	Lists all database roles. `openGauss=# \du`

Parameter	Description	Value Range	Example
\dE[S+] [PATTERN] \di[S+] [PATTERN] \ds[S+] [PATTERN] \dt[S+] [PATTERN] \dv[S+] [PATTERN]	In this group of commands, the letters E, i, s, t, and v stand for foreign table, index, sequence, table, and view, respectively. You can specify any or a combination of these letters sequenced in any order to obtain an object list. For example, **\dit** lists all indexes and tables. If + is added to the end of a command name, the physical size and related description of each object are also listed.	If **PATTERN** is specified, only objects whose names match **PATTERN** are displayed. By default, only objects you created are displayed. You can specify **PATTERN** or **S** to view other system objects.	Lists all indexes and views. `openGauss=# \div`
\dx[+] [PATTERN]	Lists installed extensions.	If **PATTERN** is specified, only extensions whose names match **PATTERN** are displayed.	Lists installed extensions. `openGauss=# \dx`
\l[+]	Lists the names, owners, character set encodings, and permissions of all the databases in the server.	-	List the names, owners, character set encodings, and permissions of all the databases in the server. `openGauss=# \l`

\sf[+] FUNCNAME	Displays the definition of a function.
	NOTE
	If the function name contains parentheses, enclose the function name with double quotation marks and add the parameter type list following the double quotation marks. Also enclose the list with parentheses.
		-	Assume a function **function_a** and a function **func()name**. This parameter will be as follows:
			`openGauss=# \sf function_a`
			`openGauss=# \sf "func()name"(argtype1, argtype2)`

Parameter	Description	Value Range	Example
\z [PATTERN]	Lists all tables, views, and sequences in the database and their access permissions.	If a pattern is given, it is a regular expression, and only matched tables, views, and sequences are shown.	Lists all tables, views, and sequences in the database and their access permissions. openGauss=# \z

Table 1-15 Description of permissions

Parameter	Description
r	**SELECT**: allows users to read data from specified tables and views.
w	**UPDATE**: allows users to update columns for specified tables.
a	**INSERT**: allows users to insert data to specified tables.
d	**DELETE**: allows users to delete data from specified tables.
D	**TRUNCATE**: allows users to delete all data from specified tables.
x	**REFERENCES**: allows users to create foreign key constraints.
t	**TRIGGER**: allows users to create a trigger on specified tables.
X	**EXECUTE**: allows users to use specified functions and the operators that are realized by the functions.
U	**USAGE**: ● For procedural languages, allows users to specify a procedural language when creating a function. ● For schemas, allows users to access objects included in specified schemas. ● For sequences, allows users to use the nextval function.
C	**CREATE**: ● For databases, allows new schemas to be created within the database. ● For schemas, allows users to create objects in a schema. ● For tablespaces, allows users to create tables in a tablespace and set the tablespace to default one when creating databases and schemas.

Parameter	Description
c	**CONNECT**: allows users to connect to specified databases.
T	**TEMPORARY**: allows users to create temporary tables.
A	**ALTER**: allows users to modify the attributes of a specified object.
P	**DROP**: allows users to delete specified objects.
m	**COMMENT**: allows users to define or modify comments of a specified object.
i	**INDEX**: allows users to create indexes on specified tables.
v	**VACUUM**: allows users to perform ANALYZE and VACUUM operations on specified tables.
*	Authorization options for preceding permissions.

Table 1-16 Formatting meta-commands

Parameter	Description
\a	Switches between aligned and unaligned table output formats.
\C [STRING]	Sets the title of any table being printed as the result of a query or cancels such a setting.
\f [STRING]	Sets the field separator for unaligned query outputs.
\H	• If the text format schema is used, switches to the HTML format. • If the HTML format schema is used, switches to the text format.
\pset NAME [VALUE]	Sets options affecting the output of query result tables. For details about the value of **NAME**, see **Table 1-17**.
\t [on|off]	Switches the display of output name information and row count footer.
\T [STRING]	Specifies attributes to be placed within the table tag in HTML output format. If this parameter is empty, no attribute is specified.
\x [on|off|auto]	Switches expanded table formatting mode.

Table 1-17 Adjustable printing options

Option	Description	Value Range
border	The value must be a number. In general, the larger the number, the more borders and lines the tables will have, but this depends on the particular format.	• The value is an integer greater than 0 in HTML format. • The value range in other formats is as follows: – 0: no border – 1: internal dividing line – 2: table frame
expanded (or x)	Switches between regular and expanded formats.	• When the expanded format is enabled, query results are displayed in two columns, with the column name on the left and the data on the right. This format is useful if the data does not fit the screen in the normal "horizontal" format. • Use the expanded format when the query output format is wider than the screen in regular format. The regular format is effective only in the aligned and wrapped formats.
fieldsep	Specifies the field separator to be used in unaligned output mode. In this way, you can create tab- or comma-separated output required by other programs. To set a tab as field separator, type **\pset fieldsep '\t'**. The default field separator is a vertical bar (|).	-
fieldsep_zero	Sets the field separator to use in unaligned output format to a zero byte.	-
footer	Switches the display of the default footer.	-

Option	Description	Value Range
format	Selects the output format. Unique abbreviations are allowed (this indicates that one letter is enough).	Value range: • **unaligned**: Write all columns of a row on one line, separated by the currently active column separator. • **aligned**: This format is standard and human-readable. • **wrapped**: This format is similar to **aligned**, but includes the packaging cross-line width data value to suit the width of the target field output. • **html**: This format outputs tables to the markup language for a document. The output is not a complete document. • **latex**: This format outputs tables to the markup language for a document. The output is not a complete document. • **troff-ms:** This format outputs tables to the markup language for a document. The output is not a complete document.
null	Sets a character string to be printed in place of a null value.	The default is to print nothing, which can be easily mistaken for an empty string.
numericlocale	Switches the display of a locale-aware character to separate groups of digits to the left of the decimal marker.	• **on**: The specified separator is displayed. • **off**: The specified separator is not displayed If this parameter is ignored, the default separator is displayed.

Option	Description	Value Range
pager	Controls the use of a pager for query and **gsql** help outputs. If the **PAGER** environment variable is set, the output is redirected to the specified program. Otherwise, the platform-dependent default value is used.	• **on**: The pager is used for terminal output that does not fit the screen. • **off**: The pager is not used. • **always**: The pager is used for all terminal output regardless of whether it fits the screen.
recordsep	Specifies the record separator to use in unaligned output mode.	-
recordsep_zero	Sets the record separator to use in unaligned output format to a zero byte.	-
tableattr (or T)	Specifies attributes to be placed inside the HTML table tag in HTML output format (such as cellpadding or bgcolor). Note that you do not need to specify **border** here because it has been used by **\pset border**. If no value is given, the table attributes do not need to be set.	-
title	Sets the table title for any subsequently printed tables. This can be used to give your output descriptive tags. If no value is given, the title does not need to be set.	-
tuples_only (or t)	Enables or disables the tuples-only mode. Full display may show extra information, such as column headers, titles, and various footers. In tuples-only mode, only the table data is shown.	-
feedback	Specifies whether to output the number of result lines.	-

Table 1-18 Connection meta-commands

Parameter	Description	Value Range
\c[onnect] [DBNAME|- USER|- HOST|- PORT|-]	Connects to a new database. If a database name contains more than 63 bytes, only the first 63 bytes are valid and are used for connection. However, the database name displayed in the command line of **gsql** is still the name before the truncation. **NOTE** If the database login user is changed during reconnection, you need to enter the password of the new user. The maximum length of the password is 999 bytes, which is restricted by the maximum value of the GUC parameter **password_max_length**.	-
\encoding [ENCODING]	Sets the client character set encoding.	Without an argument, this command shows the current encoding.
\conninfo	Prints information about the current connected database.	-

Table 1-19 OS meta-commands

Parameter	Description	Value Range
\cd [DIR]	Changes the current working directory.	An absolute path or relative path that meets the OS path naming convention
\setenv NAME [VALUE]	Sets the **NAME** environment variable to **VALUE**. If **VALUE** is not provided, do not set the environment variable.	-
\timing [on|off]	Displays how long each SQL statement takes, in milliseconds.	• **on**: specifies that the display is enabled. • **off**: indicates that the display is disabled.
\! [COMMAND]	Escapes to a separate Unix shell or runs a Unix command.	-

Table 1-20 Variable meta-commands

Parameter	Description
\prompt [TEXT] NAME	Prompts the user to use texts to specify a variable name.
\set [NAME [VALUE]]	Sets the *NAME* internal variable to **VALUE**. If more than one value is provided, *NAME* is set to the concatenation of all of them. If no second argument is given, the variable is just set with no value. Some common variables are processed differently in **gsql** and they are combinations of uppercase letters, numbers and underscores. **Table 1-21** describes a list of variables that are processed in a way different from other variables.
\unset NAME	Deletes the variable name of **gsql**.

Table 1-21 Common **\set** commands

Command	Description	Value Range
\set VERBOSITY value	This variable can be set to **default**, **verbose**, or **terse** to control redundant lines of error reports.	Value range: **default**, **verbose**, and **terse**
\set ON_ERROR_STOP value	If this variable is set, the script execution stops immediately. If this script is invoked from another script, that script will be stopped immediately as well. If the outermost script is invoked using the **-f** option rather than from an interactive gsql session, gsql will return error code **3**, indicating the difference between the current error and critical errors. (The error code for critical errors is **1**.)	Value range: **on/off**, **true/false**, **yes/no**, and **1/0**

Command	Description	Value Range
\set AUTOCOMMIT [on|off]	Sets the auto commit behavior of the current gsql connection. **on** indicates that auto commit is enabled, and **off** indicates that auto commit is disabled. By default, the gsql connection is automatically committed, and each individual statement is implicitly committed. If auto commit is disabled for performance or other purposes, you need to explicitly run the COMMIT command to ensure that transactions are committed. For example, execute the COMMIT statement to explicitly commit transactions after a specified service SQL statement is executed. Particularly, ensure that all transactions are committed before the gsql client exits. **NOTE** The auto commit is enabled in gsql by default. If you disable it, all the statements executed later will be packaged in implicit transactions, and you cannot execute statements that cannot be executed within transactions.	• **on**: The auto commit is enabled. • **off**: The auto commit is disabled.

Command	Description	Value Range
\set RETRY [retry_times]	Determines whether to enable the retry function if statement execution encounters errors. The parameter **retry_times** specifies the maximum number of retry times and the default value is **5**. Its value ranges from **5** to **10**. If the retry function has been enabled, when you run the **\set RETRY** command again, the retry function will be disabled. The configuration file **retry_errcodes.conf** shows a list of errors. If these errors occur, retry is required. This configuration file is placed in the same directory as that for executable programs. This configuration file is configured by the system rather than by users and cannot be modified by the users. The retry function can be used in the following error scenarios: ● **YY002**: TCP communication errors. Print information: **Connection reset by peer**. (reset between DNs) ● **YY003**: Lock timeout. Print information: **Lock wait timeout...**/**wait transaction** *xxx* **sync time exceed** *xxx*. ● **YY004**: TCP communication errors. Print information: **Connection timed out**. ● **YY005**: Failed to issue SET commands. Print information: **ERROR SET query**. ● **YY006**: Failed to apply for memory. Print information: **memory is temporarily unavailable**. ● **YY007**: Failed to allocate memory for the libcomm library. Print information: **Memory allocate error**. ● **YY008**: No data is received during libcomm communication. Print information: **No data in buffer**. ● **YY009**: A memory allocation error occurs during libcomm communication and the connection is closed. Print information: **Close because releases memory**. ● **YY010**: The control channel fails to send or receive data during libcomm communication. Print information: **control channel is disconnect**. ● **YY011**: The data channel fails to send or receive data during libcomm	Value range of **retry_times**: **5** to **10**

Command	Description	Value Range
	communication. Print information: **data channel is disconnect**. • **YY012**: The peer connection is closed abnormally during libcomm communication. Print information: **Stream closed by remote**. • **YY013**: Failed to receive data during libcomm communication. Print information: **Wait poll unknown error**. If an error occurs, **gsql** queries connection status of all DNs. If the connection status is abnormal, **gsql** sleeps for 1 minute and tries again. In this case, the retries in most of the primary/standby switchover scenarios are involved. **NOTE** 1. Statements in transaction blocks cannot be retried upon a failure. 2. Retry is not supported if errors are found using ODBC or JDBC. 3. For SQL statements with unlogged tables, the retry is not supported if a node is faulty. 4. For **gsql** client faults, the retry is not supported.	

Table 1-22 Large object meta-commands

Parameter	Description
\lo_list	Displays a list of all GaussDB large objects stored in the database, along with the comments provided for them.

PATTERN

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

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

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

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

Senior users can use regular-expression notations, such as character classes. For example [0-9] can be used to match any digit. All regular-expression special characters work as specified in POSIX. The following characters are excluded:

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

You can write ?, (*R*+|), (*R*|), and *R* to the following pattern characters: ., *R**, and *R*?. The dollar sign ($) does not need to be used as a regular expression character because **PATTERN** must match the entire name instead of being interpreted as a regular expression (in other words, $ is automatically appended to **PATTERN**). If you do not expect a pattern to be anchored, write an asterisk (*) at its beginning or end. All regular-expression special characters within double quotation marks lose their special meanings and are matched literally. Regular-expression special characters in operator name patterns (such as the **\do** parameter) are also matched literally.

1.6 Troubleshooting

Low Connection Performance

- **log_hostname** is enabled, but DNS is incorrect.

  Connect to the database, and run **show log_hostname** to check whether **log_hostname** is enabled in the database.

  If it is enabled, the database kernel will use DNS to check the name of the host where the client is deployed. If the host where the database is configured with an incorrect or unreachable DNS server, the database connection will take a long time to set up. For details about this parameter, see the

description of **log_hostname** in section "GUC Parameter Description > Error Reports and Logs > Log Content" in the *Developer Guide*.

- The database kernel slowly runs the initialization statement.

  Problems are difficult to locate in this scenario. Try using the **strace** Linux trace command.

  ```
  strace gsql -U MyUserName -d gaussdb -h 127.0.0.1 -p 23508 -r -c '\q'
  Password for MyUserName:
  ```

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

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

  It can be determined that the database executes the SELECT VERSION() statement slowly.

  After the database is connected, you can run the **explain performance select version()** statement to find the reason why the initialization statement was run slowly. For more information, see "SQL Optimization > Introduction to the SQL Execution Plan" in *Developer Guide*.

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

- TCP connection is set up slowly.

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

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

  Or,

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

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

Problems in Setting Up Connections

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

  This problem occurs generally because an unreachable IP address or port number was specified. Check whether the values of **-h** and **-p** parameters are correct.

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

  This problem occurs generally because an incorrect username or password was entered. Contact the database administrator to check whether the username and password are correct.

- gsql: FATAL: Forbid remote connection with trust method!

  For security purposes, remote login in trust mode is forbidden. In this case, you need to modify the connection authentication information in the **pg_hba.conf** file. For details, contact the administrator.

> **NOTE**
>
> Do not modify the configurations of database hosts in the **pg_hba.conf** file. Otherwise, the database may become faulty. It is recommended that service applications be deployed outside the database instead of inside the database.

- The DN can connect to the database if **-h 127.0.0.1** is specified, and the connection will fail if **-h 127.0.0.1** is removed.

  Run the SQL statement **show unix_socket_directory** to check whether the **unix socket directory** used by the DN is the same as that specified by the environment variable **$PGHOST** in the **shell** directory.

  If they are different, set **$PGHOST** to the directory specified by **unix_socket_directory**.

  For more information about **unix_socket_directory**, see "GUC Parameter Description > Connection and Authentication > Connection Settings" in the *Developer Guide*.

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

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

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

  This problem occurs because the version of **libpq.so** used in the environment does not match that of **gsql** (or the PostgreSQL **libpq.so** exists in the environment). Run the **ldd gsql** command to check the version of the loaded **libpq.so**, and then load correct **libpq.so** by modifying the environment variable **LD_LIBRARY_PATH**.

- gsql: connect to server failed: Connection timed out

  Is the server running on host "xx.xxx.xxx.xxx" and accepting TCP/IP connections on port xxxx?

  This problem is caused by network connection faults. Check the network connection between the client and the database server. If you cannot ping from the client to the database server, the network connection is abnormal. Contact network management personnel for troubleshooting.

  ```
  ping -c 4 10.10.10.1
  PING 10.10.10.1 (10.10.10.1) 56(84) bytes of data.
  From 10.10.10.1: icmp_seq=2 Destination Host Unreachable
  From 10.10.10.1 icmp_seq=2 Destination Host Unreachable
  From 10.10.10.1 icmp_seq=3 Destination Host Unreachable
  From 10.10.10.1 icmp_seq=4 Destination Host Unreachable
  --- 10.10.10.1 ping statistics ---
  4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 2999ms
  ```

- gsql: FATAL: permission denied for database "gaussdb"

  DETAIL: User does not have CONNECT privilege.

  This problem occurs because the user does not have the permission to access the database. To solve this problem, perform the following steps:

  a. Connect to the database as the system administrator **dbadmin**.

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

  b. Grant the user with the permission to access the database.

     ```
     GRANT CONNECT ON DATABASE gaussdb TO user1;
     ```


> **NOTE**
>
> Actually, some common misoperations may also cause a database connection failure, for example, entering an incorrect database name, username, or password. In this case, the client tool will display the corresponding error messages.

```
gsql -d gaussdb -p 8000
gsql: FATAL:  database "gaussdb" does not exist

gsql -d gaussdb -U user1 -p 8000
Password for user user1:
gsql: FATAL:  Invalid username/password, login denied.
```

- gsql: FATAL: sorry, too many clients already, active/non-active: 197/3.

  This problem occurs because the number of system connections exceeds the allowed maximum. Contact the DBA database administrator to release unnecessary sessions.

  You can check the number of connections as described in **Table 1-23**.

  You can view the session status in the **PG_STAT_ACTIVITY** view. To release unnecessary sessions, use the **pg_terminate_backend** function.

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

  The value of **pid** is the thread ID of the session. Terminate the session using its thread ID.

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

  If a command output similar to the following is displayed, the session is successfully terminated.

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

Table 1-23 Viewing the number of session connections

Description	Command

View the maximum number of sessions connected to a specific user.	Run the following command to view the upper limit of the number of **USER1**'s session connections. **-1** indicates that no upper limit is set for the number of **USER1**'s session connections.
	`SELECT ROLNAME,ROLCONNLIMIT FROM PG_ROLES WHERE ROLNAME='user1';`
	`rolname | rolconnlimit`
	`---------+--------------`
	`user1 | -1`
	`(1 row)`

Description	Command
View the number of session connections that have been used by a specified user.	Run the following command to view the number of session connections that have been used by **USER1**. **1** indicates the number of session connections that have been used by **USER1**. SELECT COUNT(*) FROM dv_sessions WHERE USERNAME='user1'; count ------- 1 (1 row)
View the maximum number of sessions connected to a specific database.	Run the following command to view the upper limit of the number of **gaussdb**'s session connections. **-1** indicates that no upper limit is set for the number of **gaussdb**'s session connections. SELECT DATNAME,DATCONNLIMIT FROM PG_DATABASE WHERE DATNAME='gaussdb'; datname | datconnlimit ----------+-------------- gaussdb | -1 (1 row)
View the number of session connections that have been used by a specific database.	Run the following command to view the number of session connections that have been used by **gaussdb**. **1** indicates the number of session connections that have been used by **gaussdb**. SELECT COUNT(*) FROM PG_STAT_ACTIVITY WHERE DATNAME='gaussdb'; count ------- 1 (1 row)
View the number of session connections that have been used by all users.	Run the following command to view the number of session connections that have been used by all users: SELECT COUNT(*) FROM dv_sessions; count ------- 10 (1 row)

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

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

  Generally, this problem is caused by the incorrect host and port (that is, the *xxx* part in the error information) specified by the **-h** and **-p** parameters. As a result, the communication fails. Occasionally, this problem is caused by network faults. To resolve this problem, check whether the host name and port number of the database are correct.
- gsql: could not receive data from server: Connection reset by peer.

Check whether DN logs contain information similar to "FATAL: cipher file "/data/coordinator/server.key.cipher" has group or world access". This error is usually caused by incorrect tampering with the permissions for data directories or some key files. For details about how to correct the permissions, see related permissions for files on other normal instances.

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

  In **pg_hba.conf** of the target DN, the authentication mode is set to **gss** for authenticating the IP address of the current client. However, this authentication algorithm cannot authenticate clients. Change the authentication algorithm to **sha256** and try again. For details, contact the administrator.

  NOTE

  - Do not modify the configurations of database hosts in the **pg_hba.conf** file. Otherwise, the database may become faulty.
  - It is recommended that service applications be deployed outside the database instead of inside the database.

Other Faults

- There is a core dump or abnormal exit due to the bus error.

  Generally, this problem is caused by changes in loading the shared dynamic library (.so file in Linux) during process running. Alternatively, if the process binary file changes, the execution code for the OS to load machines or the entry for loading a dependent library will change accordingly. In this case, the OS kills the process for protection purposes, generating a core dump file.

  To resolve this problem, try again. In addition, do not run service programs in a database during O&M operations, such as an upgrade, preventing such a problem caused by file replacement during the upgrade.

  NOTE

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

2 gs_loader

Overview

- gs_loader is used to import data. gs_loader converts the syntax supported by the control file to the \COPY syntax, uses the existing \COPY function to import data, and records the \COPY result in logs.
- Before using gs_loader, ensure that the gs_loader version is consistent with the gsql version and database version.

Log Level Configuration

Set the log level for developers to view. After the setting, the tool running information is printed on the console.

```
export gs_loader_log_level=debug
export gs_loader_log_level=info
export gs_loader_log_level=warning
export gs_loader_log_level=error
```

Permission

The application scenarios are classified into separation-of-duties and non-separation-of-duties scenarios. You can set **enableSeparationOfDuty** to **on** or **off** to enable or disable the separation of duties function.

- If **enableSeparationOfDuty** is set to **off**:

  The user can be a common user or the database administrator. If the user is a common user, the administrator needs to grant permissions to the common user. The administrator account can be used directly.

  Create a user.

  ```
  CREATE USER load_user WITH PASSWORD '************';
  ```

  Create related tables and grant permissions.

  ```
  GRANT ALL ON FUNCTION copy_error_log_create() TO load_user;
  GRANT ALL ON SCHEMA public TO load_user;
  SELECT copy_error_log_create();
  SELECT copy_summary_create();
  GRANT ALL PRIVILEGES ON  public.pgxc_copy_error_log To load_user;
  GRANT ALL PRIVILEGES ON  public.gs_copy_summary To load_user;
  ```

- If **enableSeparationOfDuty** is set to **on**:

The user can be a common user or the database administrator. Create the **pgxc_copy_error_log** and **gs_copy_summary** tables in their respective schemas and add indexes. No permission granting is required.

Create a user.

```
CREATE USER load_user WITH PASSWORD '********';
```

Create related tables and add indexes.

```
CREATE TABLE load_user.pgxc_copy_error_log (relname varchar, begintime timestamptz, filename varchar, lineno int8, rawrecord text, detail text);
CREATE INDEX copy_error_log_relname_idx ON load_user.pgxc_copy_error_log(relname);
CREATE TABLE load_user.gs_copy_summary(relname varchar, begintime timestamptz, endtime timestamptz, id bigint, pid bigint, readrows bigint, skiprows bigint, loadrows bigint, errorrows bigint, whenrows bigint, allnullrows bigint, detail text);
CREATE INDEX gs_copy_summary_idx ON load_user.gs_copy_summary(id);
```

Usage Environment

You need to add the tool path to *PATH*. gs_loader supports SSL encrypted communication. The method of using gs_loader is the same as that of using gsql. For details, refer to the *Administrator Guide*.

Adding System Catalogs

The **gs_copy_summary** table is added to record the COPY execution result summary, including the number of successful rows, number of error rows, number of ignored rows, and number of empty rows.

The **copy_summary_create** function is added to create the **gs_copy_summary** table.

The format of the **gs_copy_summary** table is as follows:

```
relname     | public.sqlldr_tbl
begintime   | 2021-09-03 16:00:11.7129-04
endtime     | 2021-09-03 16:00:15.259908-04
id          | 21870
pid         | 47582725060352
readrows    | 100000
skiprows    | 0
loadrows    | 111
errorrows   | 0
whenrows    | 99889
allnullrows | 0
detail      | 111 Rows successfully loaded.
            | 0 Rows not loaded due to data errors.
            | 99889 Rows not loaded because all WHEN clauses were failed.
            | 0 Rows not loaded because all fields were null.
            |
```

Usage Guidelines

Step 1 Create a user and related tables, and add indexes.

- If the separation of duties function is disabled for common users only:

  Create a user.

  ```
  CREATE USER load_user WITH PASSWORD '************';
  ```

  Create related tables and grant permissions.

  ```
  GRANT ALL ON FUNCTION copy_error_log_create() TO load_user;
  GRANT ALL ON SCHEMA public TO load_user;
  SELECT copy_error_log_create();
  ```



```
SELECT copy_summary_create();
GRANT ALL PRIVILEGES ON  public.pgxc_copy_error_log To load_user;
GRANT ALL PRIVILEGES ON  public.gs_copy_summary To load_user;
```

- If the separation of duties function is enabled for common users and administrators:

  Create a user.

  ```
  CREATE USER load_user WITH PASSWORD '********';
  ```

  Create related tables and add indexes.

  ```
  CREATE TABLE load_user.pgxc_copy_error_log (relname varchar, begintime timestamptz, filename varchar, lineno int8, rawrecord text, detail text);
  CREATE INDEX copy_error_log_relname_idx ON load_user.pgxc_copy_error_log(relname);
  CREATE TABLE load_user.gs_copy_summary(relname varchar, begintime timestamptz, endtime timestamptz, id bigint, pid bigint, readrows bigint, skiprows bigint, loadrows bigint, errorrows bigint, whenrows bigint, allnullrows bigint, detail text);
  CREATE INDEX gs_copy_summary_idx ON load_user.gs_copy_summary(id);
  ```

Step 2 Create a table and a control file, and prepare a data file.

Create the **loader_tbl** table.

```
CREATE TABLE  loader_tbl
(
   ID   NUMBER,
   NAME VARCHAR2(20),
   CON  VARCHAR2(20),
   DT   DATE
);
```

Create the **loader.ctl** control file.

```
LOAD DATA
truncate into table loader_tbl
WHEN (2:2) = ','
fields terminated by ','
trailing nullcols
(
   id integer external,
   name char(32),
   con ":id || '-' || :name",
   dt date
)
```

Create the **data.csv** data file.

```
1,OK,,2007-07-8
2,OK,,2008-07-8
3,OK,,2009-07-8
4,OK,,2007-07-8
43,DISCARD,,2007-07-8
,,,
32,DISCARD,,2007-07-8
a,ERROR int,,2007-07-8
8,ERROR date,,2007-37-8
,,,,
 ,
8,ERROR fields,,2007-37-8
,,,
5,OK,,2021-07-30
```

Step 3 Import the data.

Before importing data, ensure that the gs_loader tool has the execute permission. Ensure that the current path has the write permission on files. (The gs_loader generates some temporary files during the processing and automatically deletes them after the import is complete.)


```
gs_loader control=loader.ctl data=data.csv db=testdb bad=loader.bad errors=5 port=8000 passwd=************
user=load_user
```

Execution result:

```
gs_loader: version 0.1

 10 Rows successfully loaded.

log file is:
 loader.log
```

----End

Parameter Description

Table 2-1 gs_loader parameter description

Parameter	Parameter Description	Parameter Type: Value Range
help	Displays help information.	-
user	Database connection user (equivalent to **-U**).	String
-U	Database connection user (equivalent to **user**).	String
passwd	User password (equivalent to **-W**).	String
-W	User password (equivalent to **passwd**).	String
db	Database name. This parameter is mandatory and is equivalent to **-d**.	String
-d	Database name. This parameter is mandatory and is equivalent to **db**.	String
host	Specifies the host name of the machine on which the server is running or the directory for the Unix-domain socket (equivalent to **-h**).	See the **gsql --host** parameter.
-h	Specifies the host name of the machine on which the server is running or the directory for the Unix-domain socket (equivalent to **host**).	See the **gsql --host** parameter.
port	Port number of the database server (equivalent to **-p**).	See the **gsql --port** parameter.
-p	Port number of the database server (equivalent to **port**).	See the **gsql --port** reference.
create	Determines whether to create the **pgxc_copy_error_log** and **gs_copy_summary** tables.	The value can be **true** or **false**. The default value is **true**.

Parameter	Parameter Description	Parameter Type: Value Range
clean	Indicates whether to clear the error record.	The value can be **true** or **false**. The default value is **false**.
data	(Mandatory) Data file. You can specify multiple data files or use wildcards (*) and question marks (?) to represent multiple data files.	String
control	(Mandatory) Name of a control file.	String
log	Name of a log file.	String
bad	Name of the file recording the error lines. You can also specify a directory to generate the file based on the data file name.	String
discard	Name of the file recording the lines that fail to be matched by WHEN. You can also specify a directory to generate the file name based on the data file name.	String
errors	Maximum number of error lines in a data file.	Integer Default value: **0**
skip	Number of first lines that can be skipped in a data file.	Integer Default value: **0**
bindsize	Only syntax compatibility is implemented, but functions are not implemented.	-
rows	Only syntax compatibility is implemented, but functions are not implemented.	-

CAUTION

- All parameters are in lowercase and are compatible with the gsql login mode, including **-p** port number, **-h** host, **-d** database, **-U** username, and **-W** password.
- gs_loader uses a .bad file to record errors from the rawrecord column in an error table. The error table does not record rawrecord if an error cannot be read by certain code. In this case, a blank line is recorded in the .bad file.

Control Files

- Syntax


```
LOAD [ DATA ]
[CHARACTERSET char_set_name]
[INFILE [directory_path] [filename ] ]
[BADFILE [directory_path] [filename ] ]
[{ INSERT | APPEND | REPLACE | TRUNCATE }]
INTO TABLE table_name
[{ INSERT | APPEND | REPLACE | TRUNCATE }]
[FIELDS CSV]
[TERMINATED [BY] { 'string' }]
[OPTIONALLY ENCLOSED BY { 'string' }]
[TRAILING NULLCOLS]
[ WHEN { (start:end) | column_name } {= | !=} 'string' ]
[(
col_name [ [ POSITION ({ start:end }) ]  ["sql_string"] ] | [ FILLER [column_type [external] ] ] |
[ CONSTANT "string" ] | [ SEQUENCE ( { COUNT | MAX | integer } [, incr] ) ]|[NULLIF (COL=BLANKS)]
[, ...]
)]
```

- Parameter description:
  - **CHARACTERSET**

    Character set.

    Value range: a string.
  - **INFILE**

    The current keyword is invalid and needs to occupy a separate line in the control file. The keyword is ignored during running. You need to specify the corresponding data file in the gs_loader command line parameters.
  - **BADFILE**

    The current keyword is invalid and will be ignored during running. If badfile is not specified in the gs_loader command, a badfile will be generated based on the name of the corresponding control file.
  - **INSERT | APPEND | REPLACE | TRUNCATE**

    Import mode.

    INSERT: If the table contains data, an error is reported.

    APPEND: Data is inserted directly.

    REPLACE: If the table contains data, all data is deleted and then inserted.

    TRUNCATE: If the table contains data, all data is deleted and then inserted.
  - **table_name**

    Specifies the name (possibly schema-qualified) of an existing table.

    Value range: an existing table name
  - FIELDS csv

    Specifies that the CSV mode of COPY is used. In CSV mode, the default separator is a comma (,), and the default quotation mark is a double quotation mark (").
  - **TERMINATED [BY] { 'string' }**

    The string that separates columns within each row (line) of the file, and it cannot be larger than 10 bytes.

    Value range: The delimiter cannot include any of the following characters: \.abcdefghijklmnopqrstuvwxyz0123456789

    Value range: The default value is a tab character in text format and a comma in CSV format.

- **OPTIONALLY ENCLOSED BY { 'string' }**

  Specifies a quoted character string for a CSV file.

  The default value is double quotation marks (") only in CSV mode that is explicitly specified by the **FIELDS CSV** parameter.

  In other modes, there is no default value.

- **TRAILING NULLCOLS**

  Specifies how to handle the problem that multiple columns of a row in a source data file are lost during data import.

- **WHEN { (start:end) | column_name } {= | !=}**

  Filters rows by character string between **start** and **end** or by column name.

  Value range: a string.

- **POSITION ({ start:end })**

  Processes columns and obtain the corresponding character strings between **start** and **end**.

- **"sql_string"**

  Processes columns and calculates column values based on column expressions.

  Value range: a string.

- **FILLER**

  Processes columns. If FILLER occurs, this column is skipped.

- **CONSTANT**

  Processes columns and sets the inserted columns to constants.

  Value range: a string.

- **SEQUENCE ({ COUNT | MAX | integer } [, incr])**

  Processes columns to generate the corresponding sequence values.

  - **COUNT**: The count starts based on the number of rows in the table.
  - **MAX**: The count starts from the maximum value of this column in the table.
  - **integer**: The count starts from the specified value.
  - **incr**: indicates the increment each time.

- NULLIF

  Leave the field empty. Currently, only the COL POSITION() CHAR NULLIF (COL=BLANKS) syntax is supported.

CAUTION

- OPTIONS, INFILE, and BADFILE are not supported. Syntax errors are not reported only in specific scenarios.
- gs_loader uses a .bad file to record errors from the rawrecord column in an error table. The error table does not record rawrecord if an error cannot be read by certain code. In this case, a blank line is recorded in the .bad file.

