GaussDB
3.x

Tool Reference for Primary+Standby Instances

Issue	01
Date	2024-05-17

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

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 features for users. For details, see **Advanced Features**.

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, CREATE_CREDENTIAL, ACCESSKEY, and SECRET_KEY) 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:

  ```
  gaussdb=# \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:

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

  This variable reference method is applicable to regular SQL statements and meta-commands except \copy, \ef, \help, \sf, and \!.

  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:

```
gaussdb=# \set foo 'HR.areaS'
gaussdb=# 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]**:
		`gaussdb=> \set PROMPT1 %M`
		`[local:/tmp/gaussdba_mppdb]`
		● Change the prompt to **name**:
		`gaussdb=> \set PROMPT1 name`
		`name`
		● Change the prompt to **=**:
		`gaussdb=> \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.
		`gaussdb=# \set PROMPT2 TEST`
		`gaussdb=# 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.
		`gaussdb=# \set PROMPT3 '>>>>'`
		`gaussdb=# 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 UDS or **[local:/dir/name]** if the UDS 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 UDS.

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: `gaussdb=> \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 postgres database as user **omm**.

```
gsql -d postgres -p 8000
```

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

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

Example 3: *postgres* and *omm* do not belong to any parameter, and they are used as the database name and the username, respectively.

```
gsql postgres omm -p 8000
```

Equals

```
gsql -d postgres -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:

```
gaussdb=# \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:

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

Query the table definition:

```
gaussdb=# \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**.

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

Change the prompt.

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

Query the table:

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

```
omm@[local] gaussdb=# SELECT * FROM HR.areaS;
+---------+------------------------+
| area_id |       area_name        |
+---------+------------------------+
|       1 | Europe                 |
|       2 | Americas               |
|       3 | Asia                   |
|       4 | Middle East and Africa |
+---------+------------------------+
(4 rows)
omm@[local] gaussdb=# \pset border 0
Border style is 0.
omm@[local] gaussdb=# 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] gaussdb=# \a \t \x
Output format is unaligned.
Showing only tuples.
Expanded display is on.
omm@[local] gaussdb=# 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] gaussdb=#
```

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=<FILE_NAME>      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 Connect to the database.

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

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

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

```
gsql ((GaussDB Kernel 503.1.XXX build f521c606) compiled at 2021-09-16 14:55:22 commit 2935 last mr 
6385 release)
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.
```

----End

Step 1 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.
	`gaussdb=# \h`
	`Available help:`
	`  ABORT`
	`  ALTER AGGREGATE`
	
	`... ...`
	For example, view parameters of the **CREATE DATABASE** command:
	`gaussdb=# \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.
	`gaussdb=# \?`
	`General`
	`  \copyright             show GaussDB Kernel 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 DBNAMEs, that is, connection strings in the format of **'postgres[ql]://[user[:password]@][netloc][:port][, ...][/dbname][?param1=value1&...]'** or **'[key=value] [...]'** as DBNAMEs. gsql parses connection information from the connection strings and preferentially uses the information. **CAUTION** When gsql uses the extended DBNAME to create a connection, the **replication** parameter cannot be specified.	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 - (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.	-

Parameter	Description	Value Range
-v, --set, --variable=NAME =VALUE	Sets gsql variable *NAME* to *VALUE*. For details about variable examples and descriptions, see **Variables**.	-
-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** and **COMMIT** are added to the start and end of the script, respectively, so that the script is executed as one transaction. This ensures that the script is executed successfully. If the script cannot be executed, the script is invalid. **NOTE** If the script has used **START TRANSACTION**, **COMMIT**, 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.	-

Parameter	Description	Value Range
-E, --echo-hidden	Echoes the actual queries generated by **\d** and other backslash commands.	-
-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 callers (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 node to check data consistency between the primary and standby nodes.	-
-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, -C1, --enable-client-encryption=1	Enables the encrypted database function when the **-C** parameter is used to connect to a local or remote database. The encrypted equality query is supported.	-

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.	-
-g,	Prints all SQL statements from a file.	-

Table 1-10 Connection parameters

Parameter	Description	Value Range
-h, --host=HOSTNAME	Specifies the host name of the running server, the UDS path, or the domain name. You can specify multiple host addresses by using character strings separated by commas (,). If multiple host addresses are specified, the primary node address is automatically selected for connection by default. You can set the PGTARGETSESSIONATTRS environment variable to connect to different types of nodes. The mapping between variables and node types is as follows: **A value of the PGTARGETSESSIONATTRS environment variable indicates a type of node to be connected.** **read-write**: readable and writable node. **read-only**: read-only node. **primary** or not set: primary node. **standby**: standby node. **prefer-standby**: preferred standby node. If there is no standby node, **any** is used. **any**: The role is not checked. **NOTE** If **-h** specifies only one domain name but the domain name corresponds to multiple IP addresses, the automatic primary node selection function cannot be triggered.	If the host name is omitted, gsql connects to the server of the local host over UDS or over TCP/IP to connect to local host without UDS.
-p, --port=PORT	Specifies the port number of the database server. You can configure one or more port numbers. When one port number is configured, all host addresses use the same port for connection. When multiple port numbers are configured, the sequence is the same as the host address sequence, and the number of port numbers must be the same as the number of host addresses. If they are different, an error is reported. 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.

Parameter	Description	Value Range
-U, --username=USERNAME	Specifies the user who 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 who 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 instance, 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-commands

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 commands listed in the following table 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.	-
\g [FILE] or ;	Performs a query operation and sends the result to a file or pipe.	-

Parameter	Description	Value Range
\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.
\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. ● It is not recommended that you run **SET** statements 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.

Parameter	Description	Value Range
\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 }[LOAD] [LOAD_DISCARD 'string'] [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 only applies to small-batch data import with uniform formats. 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. ● Both the TEXT and CSV formats of \COPY support the header function. ● The LOAD function is used by gs_loader to call COPY after syntax conversion. It is not an active calling function. ● The LOAD_DISCARD function is used by gs_loader to discard file path after parsing. It is not an active calling function.
\echo [STRING]	Writes character strings to the standard output.
\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.

Parameter	Description
\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 **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 **search_path**. `gaussdb=# \d`
\d[S+] NAME	Lists the structure of specified tables, views, and indexes. 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.	-	List the structure of table **a**. `gaussdb=# \dtable+ a`
\d+ [PATTERN]	Lists all tables, views, and indexes. 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.	If **PATTERN** is specified, only tables, views, and indexes whose names match **PATTERN** are displayed.	List all tables, views, and indexes whose names start with **f**. `gaussdb=# \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. 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.	If **PATTERN** is specified, only aggregate functions whose names match **PATTERN** are displayed.	List all available aggregate functions whose names start with **f**, together with their return value types and the data types. `gaussdb=# \da f*`
\db[+] [PATTERN]	Lists all available tablespaces.	If **PATTERN** is specified, only tablespaces whose names match **PATTERN** are displayed.	List all available tablespaces whose names start with **p**. `gaussdb=# \db p*`
\dc[S+] [PATTERN]	Lists all available conversions between character-set encodings. 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.	If **PATTERN** is specified, only conversions whose names match **PATTERN** are displayed.	List all available conversions between character-set encodings. `gaussdb=# \dc *`
\dC[+] [PATTERN]	Lists all available type conversions. 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. PATTERN must be the actual type name and cannot be an alias.	If **PATTERN** is specified, only conversions whose names match **PATTERN** are displayed.	List all type conversions whose patten names start with **c**. `gaussdb=# \dC c*`

Parameter	Description	Value Range	Example
\dd[S] [PATTERN]	Lists descriptions about objects matching **PATTERN**. 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.	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.	List all visible objects. `gaussdb=# \dd`
\ddp [PATTERN]	Lists all default permissions.	If **PATTERN** is specified, only permissions whose names match **PATTERN** are displayed.	List all default permissions. `gaussdb=# \ddp`
\dD[S+] [PATTERN]	Lists all available domains. 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.	If **PATTERN** is specified, only domains whose names match **PATTERN** are displayed.	List all available domains. `gaussdb=# \dD`
\det[+] [PATTERN]	Lists all foreign tables. 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.	If **PATTERN** is specified, only tables whose names match **PATTERN** are displayed.	List all foreign tables. `gaussdb=# \det`
\des[+] [PATTERN]	Lists all foreign servers.	If **PATTERN** is specified, only servers whose names match **PATTERN** are displayed.	List all foreign servers. `gaussdb=# \des`

Parameter	Description	Value Range	Example
\deu[+] [PATTERN]	Lists all user mappings.	If **PATTERN** is specified, only information whose name matches **PATTERN** is displayed.	List all user mappings. `gaussdb=# \deu`
\dew[+] [PATTERN]	Lists all encapsulated external data.	If **PATTERN** is specified, only data whose name matches **PATTERN** is displayed.	List all encapsulated external data. `gaussdb=# \dew`
\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. 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.	If **PATTERN** is specified, only functions whose names match **PATTERN** are displayed.	List all available functions, together with their parameters and return types. `gaussdb=# \df`
\dF[+] [PATTERN]	Lists all text search configuration information. 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.	If **PATTERN** is specified, only configurations whose names match **PATTERN** are displayed.	List all text search configuration information. `gaussdb=# \dF+`
\dFd[+] [PATTERN]	Lists all text search dictionaries. 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.	If **PATTERN** is specified, only dictionaries whose names match **PATTERN** are displayed.	List all text search dictionaries. `gaussdb=# \dFd`

Parameter	Description	Value Range	Example
\dFp[+] [PATTERN]	Lists all text search analyzers. 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.	If **PATTERN** is specified, only analyzers whose names match **PATTERN** are displayed.	List all text search analyzers. `gaussdb=# \dFp`
\dFt[+] [PATTERN]	Lists all text search templates. 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.	If **PATTERN** is specified, only templates whose names match **PATTERN** are displayed.	List all text search templates. `gaussdb=# \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). `gaussdb=# \dg j?e`
\dl	This is an alias for **\lo_list**, which shows a list of large objects.	-	List all large objects. `gaussdb=# \dl`
\dL[S+] [PATTERN]	Lists all available program languages.	If **PATTERN** is specified, only languages whose names match **PATTERN** are displayed.	List all available program languages. `gaussdb=# \dL`
\dm[S+] [PATTERN]	Lists materialized views. 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.	If **PATTERN** is specified, only materialized views whose names match **PATTERN** are displayed.	List materialized views. `gaussdb=# \dm`

Parameter	Description	Value Range	Example
\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 the pattern are shown. By default, only schemas you created are displayed.	List information about all schemas whose names start with **d**. `gaussdb=# \dn+ d*`
\do[S] [PATTERN]	Lists available operators with their operand and return types. 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.	If **PATTERN** is specified, only operators whose names match **PATTERN** are displayed. By default, only the operators created by the user are listed.	List available operators with their operand and return types. `gaussdb=# \do`
\dO[S+] [PATTERN]	Lists collation rules. 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.	If **PATTERN** is specified, only rules whose names match **PATTERN** are displayed. By default, only user-created rules are shown.	List collation rules. `gaussdb=# \dO`

Parameter	Description	Value Range	Example

\dp [PATTERN]	Lists tables, views, and related permissions. 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.
	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 the pattern are shown.	List tables, views, and related permissions.
			`gaussdb=# \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.	List all the modified configuration parameters of the database. `gaussdb=# \drds * dbname`
\dT[S+] [PATTERN]	Lists all data types. 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.	If **PATTERN** is specified, only types whose names match **PATTERN** are displayed.	List all data types. `gaussdb=# \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.	List all database roles. `gaussdb=# \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. 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. 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.	List all indexes and views. `gaussdb=# \div`
\dx[+] [PATTERN]	Lists installed extensions.	If **PATTERN** is specified, only extensions whose names match **PATTERN** are displayed.	List installed extensions. `gaussdb=# \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. `gaussdb=# \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:
			`gaussdb=# \sf function_a`
			`gaussdb=# \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. `gaussdb=# \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 unsets any such title.
\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 mode is useful if the data does not fit on the screen in the normal "horizontal" mode. ● 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.
numericlo cale	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]	Toggles a display of how long each SQL statement takes, in milliseconds (exclude the time of screen displaying).	• **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 called from another script, that script will be stopped immediately as well. If the primary script is called using the **-f** option rather than from one gsql session, gsql will return error code **3**, indicating the difference between the current error and critical errors. (The error code for critical errors is **1**.)	Value range: **on/off**, **true/false**, **yes/no**, or **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.

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.

DELIMITER

The DELIMITER command is used to change the delimiter between SQL statements. The default delimiter is a semicolon (;).

Using the DELIMITER command, you can set a delimiter for the client. When a delimiter is set, the gsql client sends the SQL statements to the server for execution immediately after identifying the delimiter. However, the server still considers the semicolon (;) as the SQL statement delimiter and processes the SQL statements accordingly.

Precautions:

- Currently, delimiters cannot be set freely. The terminator can be a combination of uppercase and lowercase letters or a combination of special characters (**~ ! @ # ^ & ` ? + - * / % < > =**). The common delimiter is **//**.

- The combination of special characters should be unambiguous. Ambiguous combinations, such as comment characters ***** and **--** and combinations ending with a plus sign (+) or minus sign (-), cannot be used for delimiter naming.
- The delimiter length ranges from 0 to 15.
- The level of the terminator is session-level. When the database is switched, **delimiter_name** is set to the default value semicolon (;).
- To use other combinations, you can add quotation marks (for example, **delimiter "adbc $$"**). The quotation marks are required in statements, for example, **select 1"adbc $$"**.
- The delimiter is supported only when **sql_compatibility** is set to **'B'**.

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, the database connection will take a long time to set up. For more details about **log_hostname**, see the section "GUC Parameters".
- 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 indicates that the **SELECT VERSION()** statement was run slowly.

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

- 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**.

  NOTE

  Modify the **LD_LIBRARY_PATH** environment variable by referring to the example command provided below. In this example, **${path_to_correct_libpq_dir}** indicates the directory where **libpq.so** is located.

  ```
  export LD_LIBRARY_PATH=${path_to_correct_libpq_dir}:$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, where **-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)`

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**, where **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, where **-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**, where **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)`

Description	Command

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. Contact the administrator for the specific operations.

  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.

Installation and Deployment

Install and configure the gs_loader client tool on the server where source data files are stored so that you can use the gs_loader tool to import data.

Step 1 Create a directory for storing the gs_loader tool package.

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

Step 2 Upload the gsql package to the created directory.

Upload the gsql tool package **GaussDB-Kernel_***Database version number_OS version number***_64bit_gsql.tar.gz** (the EulerOS tool package is used as an example) in the software installation package to the directory created in the previous step.

Step 3 Go to the new directory and decompress the package.

```
cd /opt/bin
tar -zxvf GaussDB-Kernel_Database version number_OS version number_64bit_gsql.tar.gz
source gsql_env.sh
```

Step 4 Verify the tool location and version information.

```
which gs_loader
```

Step 5 Verify the client version information.

The gs_loader tool version corresponds to the gsql tool version. You can directly query the gsql client version.

```
gsql -V
```

Step 6 Verify that the database version is the same as the client tool version.

Use gsql to connect to the database and run the following command:

```
select version();
```

----End

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.

The **enable_copy_error_log** GUC parameter specifies whether to use the error table pgxc_copy_error_log. The default value is **off**, indicating that the error table is not used and error records are directly recorded in the .bad file of gs_loader. If this parameter is set to **on**, the error table pgxc_copy_error_log is used and error records are written to both the .bad file and error table.

- By default, if **enableSeparationOfDuty** is set to **off**, the user can be a common database user or an 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.
  a. Create a user (as an administrator).
     ```
     CREATE USER load_user WITH PASSWORD '************';
     ```
  b. Grant the public schema permission to the user (as an administrator).
     ```
     GRANT ALL ON SCHEMA public TO load_user;
     ```
  c. Create the gs_copy_summary table and grant table permissions to the user (as an administrator).
     ```
     SELECT copy_summary_create() WHERE NOT EXISTS(SELECT * FROM pg_tables WHERE
     schemaname='public' AND tablename='gs_copy_summary');
     GRANT ALL PRIVILEGES ON  public.gs_copy_summary To load_user;
     ```
  d. (Optional) Create the pgxc_copy_error_log table and grant table permissions to the user (as an administrator).

     NOTE

     If the GUC parameter **enable_copy_error_log** is not set (**off** by default) or is set to **off**, you do not need to use the error table and do not need to create it. Otherwise, you need to create the error table.
     ```
     SELECT copy_error_log_create() WHERE NOT EXISTS(SELECT * FROM pg_tables WHERE
     schemaname='public' AND tablename='pgxc_copy_error_log');
     GRANT ALL PRIVILEGES ON  public.pgxc_copy_error_log To load_user;
     ```
- If **enableSeparationOfDuty** is set to **on**, the user can be a common database user or an administrator. Create the pgxc_copy_error_log and gs_copy_summary tables in their respective schemas and add indexes. No permission granting is required.
  a. Create a user (as the initial user).
     ```
     CREATE USER load_user WITH PASSWORD '********';
     ```

b. Switch to the **load_user** user (as the initial user).

```
\c - load_user
```

c. Create the gs_copy_summary table and add an index (as the created user).

```
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);
```

d. (Optional) Create the pgxc_copy_error_log table and add an index (as the created user).

NOTE

If the GUC parameter **enable_copy_error_log** is not set (**off** by default) or is set to **off**, you do not need to use the error table and do not need to create it. Otherwise, you need to create the error table.

```
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);
```

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.

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.
            |
```

Columns in the gs_copy_summary System Catalog

Table 2-1 gs_copy_summary columns

Column	Description
relname	Name of the target table to be imported.
begintime	Start time of an import task.
endtime	End time of an import task.
id	ID of the transaction to be imported.
pid	ID of the worker thread for the current import.
readrows	Total number of data rows read by the import task.
skiprows	Total number of data rows skipped in the import task.
loadrows	Number of data rows successfully imported in the current import task.
errorrows	Number of error data rows in the current import task.
whenrows	Number of data rows that violate the WHEN filter criterion in the current import task.
allnullrows	Number of data rows where all columns are empty.
detail	Summary of the import task, including the number of successfully imported rows, number of error data rows, number of rows that violate the WHEN condition, and number of blank rows.

Usage Guidelines

Step 1 (If the separation of duties function is disabled) For common users only:

1. Create a user (as an administrator).
```
CREATE USER load_user WITH PASSWORD '************';
```
2. Grant the public schema permission to the user (as an administrator).
```
GRANT ALL ON SCHEMA public TO load_user;
```
3. Create the gs_copy_summary table and grant table permissions to the user (as an administrator).
```
SELECT copy_summary_create() WHERE NOT EXISTS(SELECT * FROM pg_tables WHERE
schemaname='public' AND tablename='gs_copy_summary');
GRANT ALL PRIVILEGES ON  public.gs_copy_summary To load_user;
```
4. (Optional) Create the pgxc_copy_error_log table and grant table permissions to the user (as an administrator).

NOTE

If the GUC parameter **enable_copy_error_log** is not set (**off** by default) or is set to **off**, you do not need to use the error table and do not need to create it. Otherwise, you need to create the error table.

```
SELECT copy_error_log_create() WHERE NOT EXISTS(SELECT * FROM pg_tables WHERE
schemaname='public' AND tablename='pgxc_copy_error_log');
GRANT ALL PRIVILEGES ON  public.pgxc_copy_error_log To load_user;
```

5. Switch to another user (as an administrator).

```
\c - load_user
```

Step 2 (If the separation of duties function is enabled) For common users and administrators:

1. Create a user (as the initial user).

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

2. Switch to the **load_user** user (as the initial user).

```
\c - load_user
```

3. Create the gs_copy_summary table and add an index (as the created user).

```
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);
```

4. (Optional) Create the pgxc_copy_error_log table and add an index (as the created user).

NOTE

If the GUC parameter **enable_copy_error_log** is not set (**off** by default) or is set to **off**, you do not need to use the error table and do not need to create it. Otherwise, you need to create the error table.

```
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);
```

Step 3 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
);
```

(On the gs_loader client) Create the control file **loader.ctl**.

```
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
)
```

(On the gs_loader client) Create the GUC parameter file **guc.txt**.

```
set a_format_copy_version='s1';
```

(On the gs_loader client) Create the data file **data.csv**.

```
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 4 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

 5 Rows successfully loaded.

log file is:
 loader.log
```

----End

CAUTION

gs_copy_summary records the called COPY syntax and details. The *[badfile]*_**bad.log** file records error data and details. To prevent the error data and details recorded during the last import from being overwritten, you are advised to use different **bad** parameters for each import. The logging function using the pgxc_copy_error_log table is disabled by default. To use the error table pgxc_copy_error_log to record error data and details, enable the GUC parameter **enable_copy_error_log**. To delete data from a table, perform the TRUNCATE or DELETE operation on the table.

Parameters

Table 2-2 gs_loader parameter description

Parameter	Parameters	Parameter Type: Value Range
help	Displays help information.	-
user	Database connection user (equivalent to **-U**).	String
-U	Database connection user (equivalent to **user**).	String

Parameter	Parameters	Parameter Type: Value Range
passwd	User password (equivalent to **-W**).	String
-W	User password (equivalent to **passwd**).	String
db	Database name. This parameter is required and is equivalent to **-d**.	String
-d	Database name. This parameter is required and is equivalent to **db**.	String
host	Specifies the host name of the running server, the UDS path, or the domain name. You can specify multiple host addresses by using character strings separated by commas (,). This parameter is equivalent to **-h**. If multiple host addresses are specified, the primary node is connected by default.	See the **gsql --host** parameter.
-h	Specifies the host name of the running server, the UDS path, or the domain name. You can specify multiple host addresses by using character strings separated by commas (,). This parameter is equivalent to **host**. If multiple host addresses are specified, the primary node is connected by default.	See the **gsql --host** parameter.
port	Specifies the port number of the database server. One or more port numbers can be configured. When one port number is configured, all IP addresses use the same port for connection. When multiple port numbers are configured, the sequence is the same as the IP address sequence, and the number of port numbers must be the same as the number of IP addresses. If they are different, an error is reported. This parameter is equivalent to **-p**.	See the **gsql --port** parameter.

Parameter	Parameters	Parameter Type: Value Range
-p	Specifies the port number of the database server. One or more port numbers can be configured. When one port number is configured, all IP addresses use the same port for connection. When multiple port numbers are configured, the sequence is the same as the IP address sequence, and the number of port numbers must be the same as the number of IP addresses. If they are different, an error is reported. This parameter is equivalent to **port**.	See the **gsql --port** parameter.
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**.
clean	Specifies whether to clear the error record.	The value can be **true** or **false**. The default value is **false**.
data	(Required) Data file. You can specify multiple data files or use wildcards (*) and question marks (?) to represent multiple data files.	String
control	(Required) Name of a control file.	String
log	Name of a log file.	String
bad	Name of the file that records the error lines and details. You can also specify a directory. If you do not specify a directory, the file is generated 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.	-

Parameter	Parameters	Parameter Type: Value Range
rows	Specifies the number of rows of data to be imported before a commit.	The value is an integer in the range [1,2147483647].

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.
- When the **rows** parameter is specified, the number of commit times cannot exceed 1000. Otherwise, the performance is affected. The number of commit times is approximately equal to the number of data rows in the data file divided by the value of **rows**. If the **rows** parameter is not specified, there is no default value for **rows**. In this case, the transaction is committed only once after all data is imported to the table.
- When gs_loader sets the GUC parameter **a_format_load_with_constraints_violation** to support non-rollback upon constraint conflicts, if a table has a BEFORE/AFTER ROW INSERT trigger, a maximum of 10,000,000 rows can be committed at a time.
- gs_loader does not support statement-level triggers when the GUC parameter **a_format_load_with_constraints_violation** is set to support non-rollback upon constraint conflicts.

Control Files

- Syntax

```
LOAD [ DATA ]
[CHARACTERSET char_set_name]
[INFILE [directory_path] [filename ] ]
[BADFILE [directory_path] [filename ] ]
[OPTIONS(name=value)]
[{ 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. Currently, the value can be **'AL32UTF8'**, **'zhs16gbk'**, or **'zhs32gb18030'**.

CAUTION

The character set specified by **CHARACTERSET** in the control file must be the same as the encoding format of the file. Otherwise, an error is reported or garbled characters are displayed in the imported data.

- **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 no .bad file is specified in the gs_loader command, a .bad file will be generated based on the name of the corresponding control file.

- **OPTIONS**

  Only the **skip** and **rows** parameters take effect. **skip=***n* indicates that the first *n* records are skipped during import, and **rows=***n* indicates the number of rows to be imported before a commit. If both the command line and control file are specified, the command line has a higher priority.

- **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.

  NOTE

  - When writing a control file (.ctl), you can specify the import mode (**INSERT | APPEND | REPLACE | TRUNCATE**) before and after the INTO TABLE table_name statement. The priority is as follows: The import mode specified after the statement takes precedence over and overwrites that specified before the statement.
  - When multiple gs_loader sessions are started to concurrently import data to the same table, you are advised to use the **APPEND** mode. If you use the **INSERT**, **REPLACE**, or **TRUNCATE** mode, an import error may occur or the imported data may be incomplete.

- **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 (").

CAUTION

In the current CSV mode, quoted line feeds are considered as part of the column data.

- **table_name**

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

Value range: an existing table name

– **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 value 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.

CAUTION

- When you set **OPTIONALLY ENCLOSED BY { 'string' }**, either there is no quotation mark on the left of the data, or the quotation marks on the left and right must be an odd number but do not have to be equal.
- Currently, **OPTIONALLY ENCLOSED BY { 'string' }** is supported only in CSV mode. If **OPTIONALLY ENCLOSED BY { 'string' }** is specified, the system enters the CSV mode by default.

– **TRAILING NULLCOLS**

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

  If one or more columns at the end of a row are null, the columns are imported to the table as null values. If this parameter is not set, an error message is displayed, indicating that the error column is null. In this case, the data in this row is processed as an error.

– **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. For details, see **• Column expression**.

  Value range: a string.

– **FILLER**

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

– **column_type [external]**

  Processes the imported data according to different data types. For details, see **• Data types**.

– **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**

  Processes columns. In multi-row import scenarios, if sysdate, constant, position, or column expression is not specified after a column name, the column whose NULLIF keyword is not specified is left empty.

  Currently, only the COL POSITION() CHAR NULLIF (COL=BLANKS) syntax is supported. For details, see **• NULLIF use cases**.

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 coming 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.
- If a large number of constraint conflicts exist in the data file to be imported at a time, for example, the memory of the database server is small (for example, 32 GB) and the number of constraint conflicts exceeds 2 million rows, or the memory is greater than 128 GB and the number of constraint conflicts exceeds 10 million rows, a large amount of cache may be occupied. As a result, "ERROR: memory is temporarily unavailable" is reported and the import fails. Therefore, you are advised not to use the feature of non-rollback upon constraint conflicts.

- If the data in the .bad file is empty, refer to the source file and row number in the error table. (The code sequence is not identified, the .bad file content is not written, and only blank rows are recorded.)

```
loader=# select * from pgxc_copy_error_log;
      relname       |           begintime           | filename | lineno | rawrecord |
detail
--------------------+-------------------------------+----------+--------+-----------
+------------------------------------------------
 public.test_gsloader | 2023-02-09 09:20:33.646843-05 | STDIN    |      1 |           | invalid byte sequence
for encoding "UTF8": 0xb4
```

```
(1 row)
// In the preceding example, for the file corresponding to the loader, search for the first row of the
data text to find the source data.
```

- NULLIF use cases

```
// Create a table.
CREATE TABLE gsloader_test_nullif(
col1   varchar2(100) not null enable,
col2   number(5,0) not null enable,
col3   varchar2(200) not null enable,
col4   varchar2(34) not null enable,
col5   varchar2(750),
col6   number(20,0),
col7   varchar2(4000),
col8   varchar2(200)
);
// Data file test.csv
6007 17060072021-09-0360070001102010000000230         1        600700010000218              0
1        1         229465        3
6007 17060072021-09-0360070001102010000000299         1        600700010000282              0
1        1         230467        3
6007 17060072021-09-0360070001102010000000242         1        600700010000255              0
1        1         226400        3
6007 17060072021-09-0360070001102010000000202         1        600700010000288              0
1        1         219107        3
6007 17060072021-09-0360070001102010000000294         1        600700010000243              0
1        1         204404        3
6007 17060072021-09-0360070001102010000000217         1        600700010000270              0
1        1         226644        3
// Control file test.ctl
LOAD DATA
CHARACTERSET UTF8
TRUNCATE
INTO TABLE gsloader_test_nullif
TRAILING NULLCOLS
(COL1 POSITION(1:10) CHAR NULLIF (COL1 = BLANKS),
COL2  POSITION(11:14) CHAR NULLIF (COL2 = BLANKS),
COL3  POSITION(21:30) CHAR NULLIF (COL3 = BLANKS),
COL4  POSITION(31:40) CHAR NULLIF (COL4 = BLANKS),
COL5  sysdate,
COL6,
COL7,
COL8 POSITION(71:80) CHAR NULLIF (COL8 = BLANKS))
// Import data.
GS_LOADER -p xxx host=xxx control=test.ctl  data=test.csv -d testdb -W xxx
// Result: Imported.
loader=# SELECT * FROM gsloader_test_nullif;
    col1    | col2 |    col3    |    col4    |        col5         | col6 | col7 |   col8
------------+------+------------+------------+---------------------+------+------+-----------
 6007 17060 |  720 | 0360070001 | 1020100000 | 2023-05-17 20:32:03 |      |      | 010000218
 6007 17060 |  720 | 0360070001 | 1020100000 | 2023-05-17 20:32:03 |      |      | 010000282
 6007 17060 |  720 | 0360070001 | 1020100000 | 2023-05-17 20:32:03 |      |      | 010000255
 6007 17060 |  720 | 0360070001 | 1020100000 | 2023-05-17 20:32:03 |      |      | 010000288
 6007 17060 |  720 | 0360070001 | 1020100000 | 2023-05-17 20:32:03 |      |      | 010000243
 6007 17060 |  720 | 0360070001 | 1020100000 | 2023-05-17 20:32:03 |      |      | 010000270
(6 rows)
```

According to the data in the imported table, after the NULLIF keyword is used, the imported columns are normal except for the columns with the specified NULLIF and sysdate calculations. The imported columns without specified calculations are empty.

- Column expression

  gs_loader supports expression conversion and scenario extension for specified columns.

```
({ column_name [ data_type ] [ AS transform_expr ] } [, ...])
```

data_type specifies the data type of the column in the expression parameter. **transform_expr** specifies the target expression and returns the result value whose data type is the same as that of the target column in the table.

Example:

– The column type is not specified in the .ctl file, and the source data does not meet the column restrictions (data type and length restrictions) in the table.

```
// Create a table.
create table t_test(id int, text varchar(5));
// Data file test.csv
addf2,bbbbaaa,20220907,
// Control file test.ctl
Load Data
TRUNCATE INTO TABLE t_test
fields terminated by ','
TRAILING NULLCOLS(
id "length(trim(:id))",
text "replace(trim(:text),'bbbb','aa')"
)
// guc_param file
set a_format_copy_version='s1';
// Import data.
gs_loader -p xxx host=xxx control=test.ctl  data=test.csv -d testdb -W xxx guc_param=test_guc.txt
// Result: Imported.
select * from t_test;
 id | text
----+-------
  5 | aaaaa
(1 row)
```

– The column type is not specified in the .ctl file, and the implicit type conversion is performed. (You are advised to add compatibility parameters because the implicit type conversion is involved.)

```
// Create a table.
create table test(mes int, mes1 text, mes2 float8, mes3 timestamp with time zone, mes4
INTEGER);
// Data file
cat load_support_transform.data
1,mmoo,12.6789,Thu Jan 01 15:04:28 1970 PST,32767
2,yyds,180.883,Thu Jun 21 19:00:00 2012 PDT,32768
// Control file
cat load_support_transform.ctl
Load Data
TRUNCATE INTO TABLE test
fields terminated by ','
TRAILING NULLCOLS(
mes,
mes1 "mes1 || mes2",
mes2 "mes2 + 1",
mes3 "date_trunc('year', mes3)",
mes4
)
// guc_param file
cat test_guc.txt
set a_format_copy_version='s1';
set a_format_dev_version='s2';
set a_format_version='10c';
// Import data.
gs_loader -p xxx host=xxx control=load_support_transform.ctl data=load_support_transform.data
-d testdb -W xxx guc_param=test_guc.txt
// Result: Imported.
select * from test;
 mes |    mes1     |  mes2   |          mes3          | mes4
-----+-------------+---------+------------------------+-------
   1 | mmoo12.6789 | 13.6789 | 1970-01-01 00:00:00+08 | 32767
   2 | yyds180.883 | 181.883 | 2012-01-01 00:00:00+08 | 32768
```

- Data types

  Correspond to **column_type [external]** in the control file. During data loading, data is processed based on the data type. gs_loader classifies data types into common and special data types.

  - Common data types
    - CHAR [(length)]:

      Reads data based on a column separator and converts the value to the CHAR type. **length** indicates the maximum length of a single piece of data, in bytes. Generally, one character occupies one byte. The value can be left blank. The scenarios are as follows:

      - If a length is not declared, the value inherits the maximum length value declared by **POSITION**.
      - If a length is declared, it overwrites the maximum length declared by **POSITION**.
      - If neither **length** nor **POSITION** declares a length, the value is set based on the length between separators.
      - The priority of the length declaration is as follows: length > POSITION > separator.
      - If none of **length**, **POSITION**, and separator declares a length, the default length is 1.
      - If the actual data length exceeds the maximum value declared by **length**, an error is reported.
    - INTEGER external [(length)]:

      Reads data based on a column separator and converts the value to the INTEGER type. The rules for using **length** are the same as those described in "CHAR [(length)]."
    - FLOAT external [(length)]:

      Reads data based on a column separator and converts the value to the FLOAT type. The rules for using **length** are the same as those described in "CHAR [(length)]."
    - DECIMAL external (length):

      Reads data based on a column separator and converts the value to the DECIMAL type. The rules for using **length** are the same as those described in "CHAR [(length)]."
    - TIMESTAMP:

      Reads data based on a column separator and converts the value to the TIMESTAMP type.
    - DATE:

      Reads data based on a column separator and converts the value to the DATE type.
    - DATE external:

      Reads data based on a column separator and converts the value to the DATE type.

- SYSDATE:

  Obtains the system time when the corresponding insertion is performed in the database. The value cannot be referenced. The referenced content is the SYSDATE character string.

– Special data types

- INTEGER:

  Ignores the column separator, reads four-byte characters, saves them based on the little-endian storage logic, parses each character into a hexadecimal ASCII code value, and converts the value into a decimal number.

- SMALLINT:

  Ignores the column separator, reads two-byte characters, saves them based on the little-endian storage logic, parses each character into a hexadecimal ASCII code value, and converts the value into a decimal number.

  Example:

```
// Create a table.
create table t_spec(col1 varchar(10), col2 varchar(10));
// Data file
cat t_spec.txt
1234,5678,
// Control file
cat t_spec.ctl
Load Data
TRUNCATE INTO TABLE t_spec
fields terminated by ','
TRAILING NULLCOLS(
col1 position(2:6) integer,
col2 position(5:8) smallint
)
// guc_param file
cat test_guc.txt
set a_format_copy_version='s1';
set a_format_dev_version='s2';
set a_format_version='10c';
// Import data.
gs_loader -p xxx host=xxx control=t_spec.ctl data=t_spec.txt -d testdb -W xxx
guc_param=test_guc.txt
// Result: Imported.
select * from t_spec;
   col1    | col2
-----------+-------
 741618482 | 13612
(1 row)
```

- RAW:

  Parses each character into an ASCII code value. The backslash (\) is not used as an escape character.

  Restriction: Separators cannot be used in RAW data.

  Example:

```
// Create a table.
create table t_raw(col raw(50));
// Data file
cat t_raw.txt
12\n\x78!<~?'k^(%s)>/c[$50]
// Control file
cat t_raw.ctl
```

```
Load Data
TRUNCATE INTO TABLE t_raw
TRAILING NULLCOLS(
col position(1:50) raw
)
// guc_param file
cat test_guc.txt
set a_format_copy_version='s1';
set a_format_dev_version='s2';
set a_format_version='10c';
// Import data.
gs_loader -p xxx host=xxx control=t_raw.ctl data=t_raw.txt -d testdb -W xxx
guc_param=test_guc.txt
// Result: Imported.
select * from t_raw;
                     col
--------------------------------------------------------
 31325C6E5C783738213C7E3F276B5E282573293E2F635B2435305D
(1 row)
```

CAUTION

- In the multi-column import scenario, if the GUC parameter is not specified, some positions and separators cannot be used at the same time.
- In the multi-column import scenario, if the GUC parameter is specified, the POSITION operation cannot be used for some columns.
- In the multi-column import scenario, if common data types and special data types are used together, you need to specify POSITION for all data types.
- When importing data of a specified data type, you need to use **guc_param** to set **a_format_copy_version** for common data types and use **guc_param** to set **a_format_copy_version**, **a_format_dev_version** and **a_format_version** for special data types.
- If a column expression involves a system function, you need to use **guc_param** to set **a_format_dev_version** and **a_format_version** based on the corresponding function.
- If the data type contains **length**, the value of **length** must be set to an integer greater than 0. The special data type **RAW(length)** is used differently from common types. For example, if **POSITION** is not specified for the common type **INTEGER EXTERNAL(length)**, an error is reported when the value of **length** is less than the data length of the corresponding column in a text file (such as .csv or .txt). If the value of **length** is greater than the data length of the corresponding column in a text file (such as .txt), the result of the INTEGER EXTERNAL type is output. If **POSITION** is not specified for the special data type **RAW(length)**, the first *length* characters are read.
- If **POSITION(start:end)** is specified, the value of **start** must be set to an integer greater than 0, and the value of **end** must be greater than or equal to the value of **start**.
- During concurrent import, if multiple names of files specified by **discard** or **bad** point to files with the same name in the same directory, gs_loader stops importing the next file and reports an error. If a previous file has been imported, the file will be overwritten.

  The following error is reported:

  ```
  ERROR: An error occurred. Please check logfile.
  ```

  In the log file:

  ```
  …lock failed: Resource temporarily unavailable…
  ```

- If the column value in the control file is not empty and the column content is not used, the location of the data file is not occupied.

  For example, the control file is as follows:

  ```
  Load Data
  TRUNCATE INTO TABLE gsloader
  fields terminated by ','
  TRAILING NULLCOLS(
  id "trim(:id)",
  text "to_char(SYSDATE,'yyyymmdd')",
  gmt_create  "trim(:gmt_create)",
  create_str "trim(:create_str)"
  )
  ```

  The data file is as follows:

```
11,HELLO,33,
```

The import result is as follows:

```
loader=# select * from gsloader;
id |  text  |      gmt_create       | create_str
----+--------+---------------------+------------
11 | 2023-02-08 16:00:54 | HELLO |  33
```