GaussDB

# Tool Guide(Centralized_V2.0-8.x)

| | |
|---|---|
| **Issue** | 01 |
| **Date** | 2025-04-14 |

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

**Huawei Cloud Computing Technologies Co., Ltd.**

| | |
|---|---|
| Address: | Huawei Cloud Data Center Jiaoxinggong Road<br>Qianzhong Avenue<br>Gui'an New District<br>Gui Zhou 550029<br>People's Republic of China |
| Website: | https://www.huaweicloud.com/intl/en-us/ |

# Contents

# 1 Database Connection Tools

## 1.1 gsql for Connecting to a Database

### 1.1.1 gsql Usage Guide

#### Description

gsql, provided by GaussDB, is a database connection tool running in the command-line interface. You can use gsql to connect to the server and perform operations and maintenance on it. gsql enables users to perform basic operations on the database and provides several advanced features.

**Basic functions:**

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

  NOTE

  If gsql is used to connect to a database, the connection timeout interval 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 **FAQ**.

- Run SQL statements: You can interactively enter SQL statements and run them, or run specified SQL statements in a file.
- Run meta-commands: Meta-commands can help administrators 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 functions:**

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

**Table 1-1** Advanced gsql features

| Feature | 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:<br>`\set varname value`<br>To delete the variables set by the **\set** command, run the following command:<br>`\unset varname`<br>**NOTE**<br>• A variable is a simple name-value pair. The value can be any characters in any length.<br>• Variable names must consist of case-sensitive letters (including non-Latin letters), digits, and underscores (_).<br>• If the **\set** *varname* meta-command (without the second parameter) is used, the variable is set without a value specified.<br>• If the **\set** meta-command without parameters is used, values of all variables are displayed.<br>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.<br>For details about examples and descriptions about SQL substitution, see **• SQL substitution**. |
| Custom prompt | Prompts of gsql can be customized. Prompts can be modified by changing the reserved three variables of gsql: *PROMPT1*, *PROMPT2*, and *PROMPT3*.<br>These variables can be user-defined or set to 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.<br>**NOTE**<br>• 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.<br>• For security reasons, the records containing sensitive words (such as PASSWORD, IDENTIFIED, GS_ENCRYPT_AES128, GS_DECRYPT_AES128, GS_ENCRYPT, GS_DECRYPT, GS_ENCRYPT_BYTEA, GS_DECRYPT_BYTEA, 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. Sensitive words are case-insensitive. |

- Variables

  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.<br>• 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.<br>● 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.<br>● If this variable is set to **off**, only the query result is displayed.<br>● 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.<br>● 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.<br>**NOTE**<br>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.<br>● **ignoredups**: A line that exists in the historical record is not written to the historical record.<br>● **ignoreboth**, **none**, or other values: All the lines read in interaction mode are saved in the historical record.<br>**NOTE**<br>**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.<br>• If this variable is set to a non-numeric value, the default value is **10**.<br>• 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` | • **on**: When an error occurs in a statement in a transaction block, the system automatically rolls back to the state before the previous command is executed and continues to execute subsequent commands. This mode works by implicitly creating a SAVEPOINT before each command of a transaction block. When an error occurs, the system automatically rolls back to the previously created SAVEPOINT to cancel the impact of the failed command.<br>• **interactive**: Automatic rollback is enabled only in interactive sessions.<br>• **off** (default value): The entire transaction is terminated when an error occurs in a statement in a transaction block. |
| ON_ERROR_STOP | `\set ON_ERROR_STOP on \| off` | • **on**: The execution stops if an error occurs. In interactive mode, gsql returns the output of executed commands immediately.<br>• **off** (default value): An error, if occurring during the execution, is ignored, and the execution continues. |

| Variable | Setting Method | Description |
|---|---|---|
| PORT | `\set PORT port` | Specifies the port number of a connected database. |
| USER | `\set USER username` | Specifies the database user you are currently connected as. |
| VERBOSITY | `\set VERBOSITY terse \| default \| verbose` | This option is used to control redundant lines in error reports.<br>● **terse**: Only critical and major error texts and text locations are returned (which is generally suitable for single-line error information).<br>● **default**: Critical and major error texts and text locations, error details, and error messages (possibly involving multiple lines) are all returned.<br>● **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'
-- Run the following command to query the HR.areaS table:
gaussdb=# SELECT * FROM :foo;
 area_id |       area_name
---------+------------------------
       4 | Middle East and Africa
       3 | Asia
       1 | Europe
       2 | Americas
(4 rows)
```

  **NOTE**

  The value of a variable is copied word by word and can even contain asymmetric quotation marks or backslash commands. Therefore, the input content must be meaningful.

- Prompts

  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.<br>The default value of *PROMPT1* is **%o%R%#**. | *PROMPT1* can be used to change the prompt.<br>● Change the prompt to **[local]**:<br>`gaussdb=# \set PROMPT1 %M`<br>`local:/data/huawei/wisequery/perfadm_mppdb`<br>● Change the prompt to **name**:<br>`gaussdb=# \set PROMPT1 name`<br>`name`<br>● Change the prompt to **=**:<br>`gaussdb=# \set PROMPT1 %R`<br>`=` |
| PROMPT2 | Specifies the prompt displayed when more input is expected (when the command is not terminated with a semicolon (;) or the quotation marks ("") are not in pair). | *PROMPT2* can be used to display the prompt.<br>`gaussdb=# \set PROMPT2 TEST`<br>`gaussdb=# SELECT * FROM HR.areaS`<br>`TEST;`<br>` area_id \|       area_name`<br>`---------+--------------------`<br>`       1 \| Europe`<br>`       2 \| Americas`<br>`       4 \| Middle East and Africa`<br>`       3 \| Asia`<br>`(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.<br>`gaussdb=# \set PROMPT3 '>>>>'`<br>`gaussdb=# COPY HR.areaS FROM STDIN;`<br>`Enter data to be copied followed by a newline.`<br>`End with a backslash and a period on a line by itself.`<br>`>>>>1 aa`<br>`>>>>2 bb`<br>`>>>>\.` |

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

| Symbol | Description |
|---|---|
| %o | Database name. For the **postgres** database, **gaussdb** is displayed. For other databases, the database name is displayed. |
| %> | Replaced by the number of the port that the host is listening on. |
| %n | Replaced by the database session username. |
| %/ | Replaced by the name of the current database. |
| %~ | Similar to %/. However, the tilde (~) is used if the database is your default database. |
| %# | The number sign (#) is used if the session user is the database administrator; otherwise, the greater-than sign (>) is used. |
| %R | ● For *PROMPT1*, **=** typically, but **^** in single-line mode, or **!** if the session is disconnected from the database (which can happen if **\connect** fails).<br>● For *PROMPT2*, it is replaced by a hyphen (-), an asterisk (*), a single or double quotation mark, or a dollar sign ($), depending on whether gsql expects more input, for example, the query is not terminated, in a /* ... */ the comment, enclosed with quotation marks, or in a dollar sign extension. |
| %x | Replaced by the transaction status.<br>● An empty string when it is not in a transaction block<br>● An asterisk (*) when it is in a transaction block<br>● An exclamation mark (!) when it is in a failed transaction block<br>● A question mark (?) when the transaction status is indefinite (for example, because there is no connection). |
| %digits | Replaced by the character with the specified byte. |
| %:name | Replaced by the value of the *name* variable of gsql. |
| %command | Replaced by 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:<br>`gaussdb=# \set PROMPT1 '%[%033[1;33;40m%]%n@%/%R%[%033[0m%]%#'`<br>The output is a boldfaced (1;) yellow-on-black (33;40) prompt on VT100-compatible, color-capable terminals. |

**Environment variables:**

For details about gsql-related environment variables, see **Table 1-5**.

**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.<br>**NOTE**<br>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 the Emacs or Vi editor, the value of this parameter 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:<br>`PSQL_EDITOR_LINENUMBER_ARG = '+'`<br>`PSQL_EDITOR_LINENUMBER_ARG='--line '`<br>A plus sign (+) is used by default on Unix. |
| PSQLRC | Specifies the location of the user's .gsqlrc file. |
| SHELL | Using the **\!** command has the same effect as executing the shell command. |
| TMPDIR | Specifies the directory for storing temporary files. The default value is **/tmp**. |

## Prerequisites

- The user using gsql must have the permission to access the database.
- gsql must match the database version.

## Precautions

- You can use the **gsql_env.sh** script in the release package to set environment variables, for example, **source gsql_env.sh**.

- If environment variables have been set, the environment variables in the script take precedence after the **gsql_env.sh** script is executed.

## Format

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

## Parameters

For details about gsql parameters, see **Table 1-6**, **Table 1-7**, **Table 1-8**, and **Table 1-9**.

**Table 1-6** 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 be connected. If the database name is not specified, the default database name generated during initialization is used.<br>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.<br>**NOTE**<br>When gsql uses the extended DBNAMEs to create a connection, the **replication** parameter cannot be specified. | A string. |
| -f, --file=FILENAME | Specifies that files are used as the source of commands instead of reading commands entered interactively. After the files are processed, gsql exits. If *FILENAME* is set to a hyphen (-), then standard input is read. | An absolute path or relative path that meets the OS path naming convention. |
| -l, --list | Lists all available databases and then exits. | - |
| -v, --set=, --variable=NAME=VALUE | Sets gsql variable *NAME* to *VALUE*.<br>For details about variable examples and descriptions, see **• Variables**. | - |

| Parameter | Description | Value Range |
|---|---|---|
| -X, --no-gsqlrc | Does not read the startup file (neither the system-wide **gsqlrc** file nor the user's **~/.gsqlrc** file).<br>**NOTE**<br>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** option 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.<br>**NOTE**<br>If the script already contains "START TRANSACTION", "COMMIT", and "ROLLBACK", this parameter is invalid. | - |
| -y, --slash-command | Terminates the current statement and sends it to the kernel for execution or re-executes executed statements (excluding gsql meta-commands).<br>**NOTE**<br>This function applies to only A-compatible databases and does not support the **-c, --command** parameter.<br>In A-compatible mode, statements are separated using semicolons (;) by default. In the PL/SQL syntax, the slash (/) is used as the separator. When this parameter is used, the separator cannot be changed. | - |
| -?, --help | Displays help information about gsql command-line parameters, and exits. | - |
| -V, --version | Prints the gsql version information and exits. | - |

**Table 1-7** Input and output parameters

| Parameter | Description | Value Range |
|---|---|---|
| -a, --echo-all | Prints all input lines to the standard output as they are read.<br>**NOTE**<br>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.<br>**NOTE**<br>When this parameter is used in some SQL statements, the sensitive information, such as user password, may be disclosed. Use this parameter with caution. | - |
| -E, --echo-hidden | Echoes the actual queries generated by **\d** and other backslash commands. | - |
| -k, --with-key=KEY | Uses gsql to decrypt imported encrypted files.<br>**NOTE**<br>● 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 the input is complete.<br>● Stored procedures and functions cannot be decrypted and imported. | - |
| -L, --log-file=FILENAME | Writes normal output source and all query output into the file specified by *FILENAME*.<br>**NOTE**<br>● When this parameter is used in some SQL statements, the sensitive information, such as user password, may be disclosed. Use this parameter with caution.<br>● 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.<br>**NOTE**<br>The parameter is for developers 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. | - |

| Parameter | Description | Value Range |
|---|---|---|
| -o, --output=FILENAME | Puts all query output into the file specified by *FILENAME*. | An absolute path or relative path that meets the OS path naming convention. |
| -q, --quiet | Specifies the quiet mode. No additional information will be printed. | By default, gsql displays various information. |
| -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 also be used for canceling execution. This option is mainly used to debug scripts. **NOTE** 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. | - |
| -C, -C1, --enable-client-encryption=1 | Enables the encrypted database feature when the **-C** parameter is used to connect to a local or remote database. In this way, encrypted equality query is supported. | - |
| -C3, --enable-client-encryption=3 | Enables a memory decryption emergency channel when the **-C** parameter is used to connect to a local or a remote database. The encrypted equality query and the memory decryption emergency channel features are supported. | - |

**Table 1-8** Output format parameters

| Parameter | Parameters |
|---|---|
| -A, --no-align | Switches to unaligned output mode. The default output mode is aligned. |
| -F, --field-separator=STRING | Specifies the field separator, which is the vertical bar (\|) by default. |
| -H, --html | Enables the HTML output format. |

| Parameter | Parameters |
|---|---|
| -P, --pset=VAR[=ARG] | Specifies the print option in the **\pset** format in the command line.<br>**NOTE**<br>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 insert mode on the client. It is disabled by default. |
| -t, --tuples-only | Prints only tuples. |
| -T, --table-attr=TEXT | Specifies options to be placed within the HTML table tag.<br>Use this parameter with the **-H,--html** parameter to specify the output to the HTML format. |
| -x, --expanded | Displays in the expanded format. |
| -z, --field-separator-zero | Sets the field separator in the unaligned output mode to be blank.<br>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.<br>Use this parameter with the **-A, --no-align** parameter to switch to unaligned output mode. |
| -2, --pipeline | Uses a pipe to transmit the password. This parameter cannot be used on devices and must be used together with the **-c** or **-f** parameter. |

**Table 1-9** 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 (,), or specify an IPv6 host address.<br>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 values of the *PGTARGETSESSIONATTRS* variable are as follows:<br>**read-write**: readable and writable node.<br>**read-only**: read-only node.<br>**primary** or not set: primary node.<br>**standby**: standby node.<br>**prefer-standby**: preferred standby node. If there is no standby node, **any** is used.<br>**any**: no role check.<br>**NOTE**<br>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 local server over UDS or TCP/IP if no UDS is available. |
| -p, --port=PORT | 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.<br>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.<br>**NOTE**<br>● If this parameter is specified, you also need to enter your password for identity authentication when connecting to a database. You can enter the password interactively or use the **-W** parameter to specify a password.<br>● To connect to a database, add an escape character before any dollar sign ($) in the username. | A 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.<br>**NOTE**<br>● When the server where the primary database node is located connects to the local primary database node instance, a trust connection is used by default and this parameter is ignored.<br>● To connect to a database, add an escape character before any backslash (\) or back quote (`) in the password.<br>● 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**. | A string. |

## Examples

**Example 1**

Connect to a database and run the command.

**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. For details about the gsql parameters, see **Parameters**.

```
-- Connect to the 8000 port of the postgres database on the remote host as user jack. (Replace the
parameter values as required.)
gsql -h 10.180.123.163 -d postgres -U jack -p 8000
```

**NOTE**

- If the database name is not specified, the default database name generated during initialization is used.
- If the database username is not specified, the current OS user is used as the database username by default.
- When the preceding parameters (such as **-d** or **-U**) are not specified, if the database name is not specified in the connection command (**-d**), the parameter is interpreted as the database name; if the database name is specified (**-d**) but the database username is not specified (**-U**), this parameter is interpreted as the database username.

  For example, in the following command, **postgres** and **omm** are not options and are interpreted as the database name and username, respectively.

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

  It is equivalent to the following command:

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

**Step 2** Run an SQL statement.

```
-- Create the human_staff database.
gaussdb=#  CREATE DATABASE human_staff;
CREATE DATABASE
```

**Step 3** Execute gsql meta-commands and SQL statements.

```
-- List all databases and description information in GaussDB.
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     |
 gaussdb        | omm | SQL_ASCII | C       | C     |
(6 rows)

-- Drop the human_staff database.
gaussdb=# DROP DATABASE human_staff;
DROP DATABASE
```

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

**----End**

**Example 2**

Obtain the help information.

```
gsql --help
gsql is the GaussDB Kernel interactive terminal.

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

General options:
  -c, --command=COMMAND    run only single command (SQL or internal) and exit
  -d, --dbname=DBNAME      database name to connect to (default: "omm")
  -f, --file=FILENAME      execute commands from file, then exit
  -l, --list               list available databases, then exit
  -v, --set=, --variable=NAME=VALUE
                           set gsql variable NAME to VALUE
  -V, --version            output version information, then exit
  -X, --no-gsqlrc          do not read startup file (~/.gsqlrc)
```

```
  -1 ("one"), --single-transaction
                           execute command file as a single transaction
  -?, --help               show this help, then exit

Input and output options:
  -a, --echo-all           echo all input from script
  -e, --echo-queries       echo commands sent to server
  -E, --echo-hidden        display queries that internal commands generate
  -k, --with-key=KEY       the key for decrypting the encrypted file
  -L, --log-file=FILENAME  send session log to file
  -m, --maintenance        can connect to cluster during 2-pc transaction recovery
  -n, --no-libedit         disable enhanced command line editing (libedit)
  -o, --output=FILENAME    send query results to file (or |pipe)
  -q, --quiet              run quietly (no messages, only query output)
  -C, -C1, --enable-client-encryption=1
                           enable client encryption feature
  -C2, --enable-client-encryption=2
                           enable client sorting based client encryption feature
  -C3, --enable-client-encryption=3
                           enable Trusted Domain operation based on client encryption feature
  -s, --single-step        single-step mode (confirm each query)
  -S, --single-line        single-line mode (end of line terminates SQL command)
  -y, --slash-command      enable slash command to re-execute query in buffer

Output format options:
  -A, --no-align           unaligned table output mode
  -F, --field-separator=STRING
                           set field separator (default: "|")
  -H, --html               HTML table output mode
  -P, --pset=VAR[=ARG]     set printing option VAR to ARG (see \pset command)
  -R, --record-separator=STRING
                           set record separator (default: newline)
  -r                       if this parameter is set,use libedit
  -t, --tuples-only        print rows only
  -T, --table-attr=TEXT    set HTML table tag attributes (e.g., width, border)
  -x, --expanded           turn on expanded table output
  -z, --field-separator-zero
                           set field separator to zero byte
  -0, --record-separator-zero
                           set record separator to zero byte
  -2, --pipeline           use pipeline to pass the password, forbidden to use in terminal
                           must use with -c or -f

Connection options:
  -h, --host=HOSTNAME      database server host or socket directory (default: "/data/huawei/wisequery/
perfadm_mppdb")
                           allow multi host IP address with comma separator, Use the
                           PGTARGETSESSIONATTRS(default: "primary") variable to select the
                           IP address to be connected.
                           (PGTARGETSESSIONATTRS can be {read-write|read-only|primary|
                            standby|prefer-standby|any})
  -p, --port=PORT          database server port (default: "5432")
  -U, --username=USERNAME  database user name (default: "omm")
  -W, --password=PASSWORD  the password of specified database user

For more information, type "\?" (for internal commands) or "\help" (for SQL
commands) from within gsql, or consult the gsql section in the GaussDB Kernel
documentation.
```

## 1.1.2 Meta-Command Reference

This section describes meta-commands provided by gsql after GaussDB CLI tool is used to connect to a database. After connecting to the database, you can run the **\?** command to display the help information about all available meta-commands.

## 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 **Aweird"name**.
- The analysis of parameters stops at another unquoted backslash (\), which is considered the beginning of a new meta-command. Parameters end with two backslashes (\\) and then SQL statements can be analyzed, if any. In this way, SQL and gsql commands can be used together in a line. But in any case, the arguments of a meta-command cannot continue beyond the end of the line.
- M-compatible databases do not support the **\h** meta-command.

## Command Format and Parameters

For details about meta-commands, see **Table 1-10**, **Table 1-11**, **Table 1-12**, **Table 1-13**, **Table 1-15**, **Table 1-17**, **Table 1-18**, **Table 1-19**, **Table 1-21**, and **Table 1-22**.

NOTICE

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

**Table 1-10** Common meta-commands

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

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

| Parameter | Description | Value Range |
|---|---|---|
| \parallel [on [num]\|off] | Controls the parallel execution function.<br>● **on**: The switch is enabled and the maximum number of concurrently executed tasks is **num**. The default value of **num** is **1024**.<br>● **off**: This switch is disabled.<br>**NOTE**<br>● Parallel execution is not allowed in a running transaction and a transaction is not allowed to be started during parallel execution.<br>● Parallel execution of **\d** meta-commands is not allowed.<br>● If SELECT statements are run concurrently, the return results may be displayed randomly but acceptable; however, if a core dump or process response failure occurs, it is not acceptable.<br>● It is not recommended that you run **SET** statements in concurrent tasks because they may cause unexpected results.<br>● Creation of a temporary table is not supported. 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.<br>● When **\parallel** is executed, *num* independent gsql processes can be connected to the database server.<br>● The total duration of all **\parallel** tasks cannot exceed **session_timeout**. Otherwise, the connection may fail during concurrent execution.<br>● 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.<br>● The maximum number of connections allowed by the server is determined based on **max_connection** and the number of current connections. Set **num** based on the allowed number of connections. | - |
| \q | Exits the gsql program. This command is executed in a script file only when the script terminates. | - |

| Parameter | Description | Value Range |
|---|---|---|
| delimiter | Sets 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.<br>The default delimiter between SQL statements is a semicolon (;).<br>The terminator is set at the session level. In case of database switching, the default delimiter semicolon (;) is used.<br>**NOTE**<br>The delimiter is supported only when **sql_compatibility** is set to **'B'**. | ● 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 //.<br>● 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.<br>● The delimiter length ranges from 0 to 15 characters.<br>● 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 $$"**. |

**Table 1-11** 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 1-12** Input/Output commands

| Parameter | Description |
|---|---|
| \copy { table [ ( column_list ) ] \| ( query ) } { from \| to } { filename \| stdin \| stdout \| pstdin \| pstdout } [copy parameter] [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.<br>**NOTE**<br>● \COPY only applies to small-batch data import with uniform formats. GDS or COPY is preferred for data import.<br>● \COPY specifies the number of clients to import data to implement parallel import of data files. Currently, the value range is [1,8].<br>● Parallel import using \COPY is not supported for temporary tables, binary files, data encrypted using AES-128, COPY option containing EOL, or inside a transaction. In these cases, even if the parallel parameter is specified, a non-parallel process is performed.<br>● Both the TEXT and CSV formats of \COPY support the header function.<br>● The LOAD function is used by gs_loader to call COPY after syntax conversion. It is not an active function call.<br>● The LOAD_DISCARD function is used by gs_loader to discard file path after parsing. It is not an active function call.<br>● \COPY supports all COPY commands, which are represented by **copy parameter** in the parameter list. For details about all COPY commands, see "SQL Reference > SQL Syntax > C > COPY" in *Developer Guide*. |
| \echo [STRING] | Writes character strings to the standard output. |
| prompt [STRING] | Writes character strings to the standard output, which is equivalent to **\echo**. |
| \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. |

| Parameter | Description |
|---|---|
| \ir+ FILE KEY | Similar to **\i+**, but resolves relative path names differently. |
| \o [FILE] | Saves all query results to a file. |
| \qecho [STRING] | Writes character strings to the query output flow. |

**Table 1-13** 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. | - | List all tables, views, and sequences of all schemas in **search_path**.<br>`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**.<br>`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. | - | Lists all tables, views, and indexes whose names start with **f**.<br>`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. | - | Lists all available aggregate functions whose names start with **f**, together with their return value types and the data types.<br>`gaussdb=# \da f*` |
| \db[+] [PATTERN] | Lists all available tablespaces. If **PATTERN** is specified, only tablespaces whose names match **PATTERN** are displayed. | - | Lists all available tablespaces whose names start with **p**.<br>`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. | - | Lists all available conversions between character-set encodings.<br>`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. If **PATTERN** is specified, only conversions whose names match **PATTERN** are displayed. | PATTERN must be the actual type name and cannot be an alias. | Lists all type conversions whose patten names start with **c**.<br>`gaussdb=# \dC c*` |

| Parameter | Description | Value Range | Example |
|---|---|---|---|
| \dd[S] [PATTERN] | Displays all visible objects. 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 matched objects are displayed. The objects include aggregations, functions, operators, types, relations (tables, views, indexes, sequences, and large objects), and rules. | - | Lists all visible objects. `gaussdb=# \dd` |
| \ddp [PATTERN] | Lists all default permissions. If **PATTERN** is specified, only permissions whose names match **PATTERN** are displayed. | - | Lists 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. | - | Lists all available domains. `gaussdb=# \dD` |
| \det[+] [PATTERN] | Lists all external 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. | - | Lists all external tables. `gaussdb=# \det` |
| \des[+] [PATTERN] | Lists all external servers. If **PATTERN** is specified, only servers whose names match **PATTERN** are displayed. | - | Lists all external 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. | - | Lists all user mappings.<br>`gaussdb=# \deu` |
| \dew[+] [PATTERN] | Lists all encapsulated external data. If **PATTERN** is specified, only data whose name matches **PATTERN** is displayed. | - | Lists all encapsulated external data.<br>`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. | - | Lists all available functions, together with their parameters and return types.<br>`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. | - | Lists all text search configuration information.<br>`gaussdb=# \dF+` |

| Parameter | Description | Value Range | Example |
|---|---|---|---|
| \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. | - | Lists all text search dictionaries.<br>`gaussdb=# \dFd` |
| \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. | - | Lists all text search analyzers.<br>`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. | - | Lists all text search templates.<br>`gaussdb=# \dFt` |
| \dg[+] [PATTERN] | Lists all database roles. If **PATTERN** is specified, only roles whose names match **PATTERN** are displayed.<br>**NOTE**<br>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. | - | Lists all database roles named **j?e** (the question mark (?) indicates any character).<br>`gaussdb=# \dg j?e` |
| \dl | Alias of **\lo_list**, which shows a list of large objects. | - | List all large objects.<br>`gaussdb=# \dl` |

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

| Parameter | Description | Value Range | Example |
|---|---|---|---|
| \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 rules you created are displayed. | - | Lists collation rules. `gaussdb=# \dO` |
| \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. If **PATTERN** is specified, only tables and views whose names match the pattern are shown. | - | Lists tables, views, and related permissions. `gaussdb=# \dp` 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 1-14**. |
| \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` |

| Parameter | Description | Value Range | Example |
|---|---|---|---|
| \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. | - | Lists all data types. `gaussdb=# \dT` |
| \du[+] [PATTERN] | Lists all database roles. If **PATTERN** is specified, only roles whose names match **PATTERN** are displayed. **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. | - | Lists all database roles. `gaussdb=# \du` |

| Parameter | Description | Value Range | Example |
|---|---|---|---|
| \dE[S+] [PATTERN]<br>\di[S+] [PATTERN]<br>\ds[S+] [PATTERN]<br>\dt[S+] [PATTERN]<br>\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 a plus sign (+) is added to the end of a command name, the physical size and related description of each object are also listed. If **PATTERN** is specified, only objects whose names match **PATTERN** are displayed. By default, only objects you created are displayed. You can specify **PATTERN** or **S** to view other system objects. | - | Lists all indexes and views.<br>`gaussdb=# \div` |
| \dx[+] [PATTERN] | Lists installed extensions. If **PATTERN** is specified, only extensions whose names match **PATTERN** are displayed. | - | Lists installed extensions.<br>`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.<br>`gaussdb=# \l` |

| Parameter | Description | Value Range | Example |
|---|---|---|---|
| \sf[+] FUNCNAME | Displays the definition of a function.<br>**NOTE**<br>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.<br>If a function with the same name exists, the definitions of multiple functions are returned. | - | Assume a function **function_a** and a function **func()name**. This parameter will be as follows:<br>`gaussdb=# \sf function_a`<br>`gaussdb=# \sf "func()name"(argtype1, argtype2)` |
| \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.<br>`gaussdb=# \z` |

NOTE

- In **Table 1-13**, **S** indicates that the system object is displayed and **+** indicates that additional object information is displayed.
- **PATTERN** specifies the name of the object to be displayed. Pay attention to the following points about **PATTERN**:
  - 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 delimiter.
    - 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.

**Table 1-14** 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**:<br>● For procedural languages, allows users to specify a procedural language when creating a function.<br>● For schemas, allows users to access objects included in specified schemas.<br>● For sequences, allows users to use the nextval function. |
| C | **CREATE**:<br>● For databases, allows new schemas to be created within the database.<br>● For schemas, allows users to create objects in a schema.<br>● For tablespaces, allows users to create tables in a tablespace and set the tablespace to default one when creating databases and schemas. |
| 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. |

| Parameter | Description |
|---|---|
| v | **VACUUM**: allows users to perform ANALYZE and VACUUM operations on specified tables. |
| * | Grants the preceding permissions. |

**Table 1-15** Formatting meta-commands

| Parameter | Description | Value Range |
|---|---|---|
| \a | Switches between aligned and unaligned table output formats. | - |
| \C [STRING] | Sets the title for the table output, or cancels the title (if there is no parameter). | Any character string or null. |
| \f [STRING] | Specifies the character or string used to separate fields in unaligned output mode. | Any character string. |
| \H | Switches between standard output and HTML table output. | - |
| \pset NAME [VALUE] | Sets options affecting the output of query result tables. | For details about the value of **NAME**, see **Table 1-16**. |
| \t [on\|off] | Switches the table output format. | • **on**: Only data rows are displayed. Column titles and row counts are not displayed.<br>• **off**: The entire table, including the column titles and row counts, is displayed. |
| \T [STRING] | Specifies attributes to be placed within the table tag in HTML output format. If this parameter is left empty, no setting is performed. | - |

| Parameter | Description | Value Range |
|---|---|---|
| \x [on\|off\|auto] | Switches the display mode of the extended rows. | • **on**: The extended display mode is enabled. Each field is displayed in a separate row.<br>• **off**: The extended display mode is disabled and the standard table display mode is used.<br>• **auto**: The extended display mode is used depending on the width of the query result. |

**Table 1-16** Adjustable printing options

| Option | Description | Value Range |
|---|---|---|
| border | Controls the border style of the table output. | • The value is an integer greater than 0 in HTML format.<br>• The value range in other formats is as follows:<br>– **0**: no border. No border is displayed when the table is output.<br>– **1**: inner borders dividing columns, without outside borders.<br>– **2**: all borders, including the outside and inner borders. |

| Option | Description | Value Range |
|---|---|---|
| expanded (or x) | Switches between regular and expanded formats. | ● **on**: The extended display mode is enabled. Each field is displayed in a separate row.<br>● **off**: The extended display mode is disabled and the standard table display mode is used. The standard format is effective only in the aligned or wrapped mode.<br>● **auto**: The extended display mode is used depending on the width of the query result. |
| fieldsep | Specifies the delimiter between columns in unaligned output mode. | Any character string. Common values include **','**, **'\t'**, and **'\|'**. If this option is not set, spaces are used as delimiters by default. |
| fieldsep_zero | Sets the delimiter between columns in unaligned output mode to ASCII NUL. | - |
| footer | Sets the footer of the query result. | ● **on**: The footer information is displayed.<br>● **off**: The footer information is not displayed. |

| Option | Description | Value Range |
|---|---|---|
| format | Specifies the output format of the query result. | • **unaligned (u)**: Columns are displayed in unaligned mode and are separated by the current delimiter.<br>• **aligned (a)**: The table is displayed in aligned mode.<br>• **wrapped (w)**: The table is displayed similar to the aligned mode but automatically wraps any excessively long value.<br>• **html (h)**: The table is displayed in HTML table format.<br>• **latex (l)**: The table is displayed in LaTeX format.<br>• **troff-ms (t)**: The table is displayed in the troff ms macro format. |
| null | Specifies the display mode of NULL values. | Any character string. If this option is not set, whitespace characters are used to replace NULL values by default. |
| numericlo cale | Determines whether to use the locale of the current system when numbers are output. | • **on**: The locale settings of the current system are used for formatting when numbers are output. For example, the number 1,000.00 is displayed as 1,000.00 in English and 1.000,00 in German.<br>• **off**: The standard numeric format, such as 1000.00, is used for number output. |
| 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.<br>• **off**: The pager is not used.<br>• **always**: The pager is used for all terminal output regardless of whether it fits the screen. |

| Option | Description | Value Range |
|---|---|---|
| recordsep | Sets the record delimiter in unaligned output mode. | Any character string. Common values include the newline characters (\n), carriage return character (\r), and carriage return line feed character (\r \n). If this option is not set, the newline character (\n) is used as the record delimiter by default. |
| recordsep _zero | Sets the record delimiter between columns in unaligned output mode to ASCII NUL. | - |
| tableattr (or T) | Declares the attributes to be placed in the HTML table tag in the HTML output format. Note: If the **\pset** border command has been executed, you do not need to set the border. | For example, bgcolor=#FFFFFF. If no value is given, the table attributes are not set. |
| title | Specifies the title of the query result. | Any character string. If this option is not set, no title is displayed in the query result. |
| tuples_onl y (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 rows in the query result. | - |

**Table 1-17** 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.<br>**NOTE**<br>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 1-18** 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**: The display is enabled.<br>● **off**: The display is disabled. |
| \! [COMMAND] | Runs OS commands. If no command follows **\!**, running it will open an independent shell UI for user interaction. | - |

**Table 1-19** Variable meta-commands

| Parameter | Description |
|---|---|
| \prompt [TEXT] NAME | Prompts the user to enter a value and stores it in a gsql variable. *TEXT* indicates the text that prompts the user to enter, and *NAME* indicates the gsql variable that stores the input value. Example:<br>`gaussdb=#  \prompt 'Enter area name: ' area`<br>`Enter area name: Asia`<br>`gaussdb=#  SELECT * FROM HR.areaS WHERE area_name = :'area';`<br>`area_id \| area_name`<br>`---------+-----------`<br>`3 \| Asia`<br>`(1 row)` |
| \set [NAME [VALUE]] | Sets the internal variable *NAME* to **VALUE**. If more than one value is given, the variable value is the concatenation result of all values. If no second argument is given, the variable is just set with no value.<br>Some common variables are processed differently in gsql and they are combinations of uppercase letters, numbers and underscores.<br>**Table 1-20** describes a list of variables that are processed in a way different from other variables. |
| \unset NAME | Deletes the variable name of gsql. |

**Table 1-20** Common **\set** commands

| Command | Description | Value Range |
|---|---|---|
| \set VERBOSITY value | Controls the redundant lines in error reports. | • **default**: Critical and major error texts and text locations, error details, and error messages (possibly involving multiple lines) are all returned.<br>• **verbose**: All error information is returned.<br>• **terse**: Only critical and major error texts and text locations are returned (which is generally suitable for single-line error information). |

| Command | Description | Value Range |
|---|---|---|
| \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**.) | • **on**: The execution stops if an error occurs. In interactive mode, gsql returns the output of executed commands immediately.<br>• **off**: An error, if occurring during the execution, is ignored, and the execution continues. |
| \set AUTOCOMMIT [on\|off] | Controls the auto commit behavior of the current gsql connection. 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.<br>**NOTE**<br>The auto commit function 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**: Auto commit is enabled.<br>• **off**: Auto commit is disabled. |

**Table 1-21** 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 the large objects. |

**Table 1-22** Fully-encrypted meta-commands

| Parameter | Description |
|---|---|
| \send_token | Sends keys to the server for caching. This function is used only when the memory decryption emergency channel is enabled. This is a fully-encrypted function. |
| \st | Sends keys to the server for caching. This function is used only when the memory decryption emergency channel is enabled. This is a fully-encrypted function. |
| \clear_token | Destroys the keys cached on the server. This function is used only when the memory decryption emergency channel is enabled. This is a fully-encrypted function. |
| \ct | Destroys the keys cached on the server. This function is used only when the memory decryption emergency channel is enabled. This is a fully-encrypted function. |
| \key_info KEY_INFO | In the fully-encrypted database features, this parameter is used to set the parameters for accessing the external key manager. |

**NOTE**

Currently, fully-encrypted databases are not M-compatible.

## Examples

### Example 1

Obtain the help information.

```
-- Connect to a database. (Replace the database name and port number as required.)
gsql -d postgres -p 8000
gsql ((GaussDB Kernel XXX.X.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.

-- View all SQL statements supported by GaussDB.
gaussdb=#  \h
Available help:
  ABORT
  ALTER AGGREGATE
  ALTER APP WORKLOAD GROUP
... ...

-- View the parameters of the DROP TABLE syntax.
gaussdb=#  \h DROP TABLE
Command:     DROP TABLE
Description: remove a table
Syntax:
DROP TABLE [ IF EXISTS ]
    { [schema.]table_name } [, ...] [ CASCADE | RESTRICT ] [ PURGE ];
```

```
-- View meta-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
... ...
```

### Example 2

Use a meta-command.

```
-- Connect to a database. (Replace the database name and port number as required.)
gsql -d postgres -p 8000
gsql ((GaussDB Kernel XXX.X.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.

-- Create a table.
gaussdb=# CREATE SCHEMA HR;
CREATE SCHEMA
gaussdb=# CREATE TABLE HR.areaS(area_ID NUMBER,area_NAME VARCHAR2(25));
NOTICE:  The 'DISTRIBUTE BY' clause is not specified. Using 'area_id' as the distribution column by default.
HINT:  Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
CREATE TABLE

-- View the definition of a table.
gaussdb=#  \d HR.areaS
               Table "hr.areas"
  Column   |         Type          | Modifiers
-----------+-----------------------+-----------
 area_id   | numeric               |
 area_name | character varying(25) |


-- Insert four rows 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

-- View a table.
gaussdb=# SELECT * FROM HR.areaS;
area_id |       area_name
---------+------------------------
       4 | Middle East and Africa
       1 | Europe
       2 | Americas
       3 | Asia
(4 rows)

-- Use the \pset command to display a table in different ways.
gaussdb=# \pset border 2
Border style is 2.

gaussdb=# SELECT * FROM HR.areaS;
+---------+------------------------+
| area_id |       area_name        |
+---------+------------------------+
|       4 | Middle East and Africa |
|       1 | Europe                 |
|       2 | Americas               |
|       3 | Asia                   |
+---------+------------------------+
(4 rows)

gaussdb=# \pset border 0
Border style is 0.

gaussdb=# SELECT * FROM HR.areaS;
area_id       area_name
------- ----------------------
      4 Middle East and Africa
      1 Europe
      2 Americas
      3 Asia
(4 rows)

-- Use a meta-command.
gaussdb=# \a \t \x
Output format is unaligned.
Showing only tuples.
Expanded display is on.

gaussdb=# SELECT * FROM HR.areaS;
area_id|4
area_name|Middle East and Africa

area_id|1
area_name|Europe

area_id|2
area_name|Americas

area_id|3
area_name|Asia

gaussdb=# \a \t \x
Output format is aligned.
Tuples only is off.
Expanded display is off.

gaussdb=# SELECT * FROM HR.areaS;
area_id       area_name
------- ----------------------
      4 Middle East and Africa
      1 Europe
      2 Americas
      3 Asia
(4 rows)

gaussdb=# DROP TABLE HR.areaS;
DROP TABLE

gaussdb=# DROP SCHEMA HR;
DROP SCHEMA
```

## 1.1.3 FAQ

### Low Connection Performance

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

  Solution:

  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".

- Problem: The database kernel runs the initialization statement slowly.

  Solution:

  It is difficult to locate faults in this scenario. Try using the **strace** Linux 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. For example, if the following operations are suspended for a long time, the database executes the **SELECT VERSION()** statement slowly.

  ```
  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}])
  ```

  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, clear the data disk space of the DN.

- Problem: The TCP connection is set up slowly.

  Solution:

  Run **strace** to check whether the initialization statement is run slowly. If the following information is displayed for a long time:

  ```
  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.

- Problem: The connection is slow due to full resource load.

  Solution:

  When the CPU, memory, or I/O usage is close to 100%, the gsql connection is slow.

  – Run the **top** command to check the CPU usage. Run the **free** command to check the memory usage. Run the **iostat** command to check the I/O load. You can also check the monitor logs in the CM Agent and the monitoring records on the database O&M platform.

  – For peak load scenarios caused by a large number of slow queries in a short period of time, you can use the port specified by (*Port number of the database server* + 1) to query the pg_stat_activity view. For slow queries, you can use the system function pg_terminate_backend to kill sessions.

  – If service overloading exists for a long time (that is, there is no obvious slow query, or new queries still become slow after slow queries are killed), reduce the service load and increase database resources.

## Problems in Setting Up Connections

- Problem: The following error message is displayed: "gsql: could not connect to server: No route to host."

  Solution:

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

- Problem: The following error message is displayed: "gsql: FATAL: Invalid username/password,login denied."

  Solution:

  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.

- Problem: The following error message is displayed: "gsql: FATAL: Forbid remote connection with trust method!"

  Solution:

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

  **NOTE**

  Do not modify the configurations of database hosts in the **gs_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.

- Problem: The database can be connected from a DN by adding **-h 127.0.0.1**, but the connection fails if **-h 127.0.0.1** is removed.

  Solution:

  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 GUC parameter **unix_socket_directory**.

  For more details about **unix_socket_directory**.

- Problem: The following error message is displayed: "The "libpq.so" loaded mismatch the version of gsql, please check it."

  Solution:

  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*.

- Problem: The following error message is displayed: "gsql: symbol lookup error: xxx/gsql: undefined symbol: libpqVersionString."

  Solution:

  This problem occurs because the version of **libpq.so** used in the environment does not match that of gsql (or the PG **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*.

- Problem: The following error message is displayed: "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?

  Solution:

  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
  ```

- Problem: The following error message is displayed: "gsql: FATAL: permission denied for database "gaussdb"."

  DETAIL: User does not have CONNECT privilege.

  Solution:

  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 a database administrator.
     ```
     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;
     ```

  In addition, many common misoperations may cause users to fail to connect to the database, 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.
  ```

- Problem: The following error message is displayed: "gsql: FATAL: sorry, too many clients already, active/non-active: 197/3."

  Solution:

  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 of user sessions 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 **user1**'s connections. **-1** indicates that no upper limit is set for **USER1**'s session connections.<br>`gaussdb=# SELECT ROLNAME,ROLCONNLIMIT FROM PG_ROLES WHERE ROLNAME='user1';`<br>` rolname \| rolconnlimit`<br>`---------+--------------`<br>` user1    \|           -1`<br>`(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 connections that have been used by **user1**. **1** indicates the number of connections that have been used by **user1**.<br>`gaussdb=# SELECT COUNT(*) FROM dv_sessions WHERE USERNAME='user1';`<br>` count`<br>`-------`<br>`     1`<br>`(1 row)` |
| View the maximum number of session connections of a specific database. | Run the following command to view the upper limit of **gaussdb**'s session connections. **-1** indicates that no upper limit is set for **gaussdb**'s session connections.<br>`gaussdb=# SELECT DATNAME,DATCONNLIMIT FROM PG_DATABASE WHERE DATNAME='gaussdb';`<br>` datname  \| datconnlimit`<br>`----------+--------------`<br>` gaussdb \|           -1`<br>`(1 row)` |
| View the number of session connections that have been used by a specified database. | Run the following command to view the number of session connections that have been used by **gaussdb**. **1** indicates the number of session connections that have been used by **gaussdb**.<br>`gaussdb=# SELECT COUNT(*) FROM PG_STAT_ACTIVITY WHERE DATNAME='gaussdb';`<br>` count`<br>`-------`<br>`     1`<br>`(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:<br>`gaussdb=# SELECT COUNT(*) FROM dv_sessions;`<br>` count`<br>`-------`<br>`     10`<br>`(1 row)` |

- Problem: The following error message is displayed: "gsql: wait xxx.xxx.xxx.xxx:xxxx timeout expired."

  Solution:

  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.
- Problem: The following error message is displayed: "gsql: could not receive data from server: Connection reset by peer."

  Solution:

  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.
- Problem: The following error message is displayed: "gsql: FATAL: GSS authentication method is not allowed because XXXX user password is not disabled."

  Solution:

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

  **NOTE**

  - Do not modify the configurations of database hosts in the **gs_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

- Problem: A core dump or abnormal exit occurs due to a bus error.

  Solution:

Generally, this problem is caused by changes to the shared dynamic library (.so file in Linux) loaded 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 terminates the process for protection purposes, generating a core dump file. To resolve this problem, please 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**

Possibly, a 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 Data Import and Export Tools

## 2.1 Overview

**Table 2-1** Overview

| Tool | Description |
|---|---|
| COPY | Syntax for importing and exporting database data in batches. COPY can be used to import and export table-level data and export query result sets. For details about the best practices for COPY, see "Best Practices > Best Practices for COPY Import and Export" in *Developer Guide*. For details about all COPY commands, see "SQL Reference > SQL Syntax > C > COPY" in *Developer Guide*. |
| **\COPY** | Meta-command for importing and exporting table-level data in batches on the gsql client. Compared with the **COPY** command, the **\COPY** meta-command can import and export client data files and supports concurrent import through multiple transactions. |
| **gs_loader** | Imports table 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 client logs. |
| **gs_dump** | Tool for database-level export. To export information about a single database, you can export the definitions and data of all or some objects (such as schemas, tables, and views) in a database. |
| **gs_dumpall** | Tool for instance-level export. You can export object definitions and data of all databases in an instance, including data of the default database, data of user-defined databases, and global objects of all databases. gsql is required for instance-level restoration. |

| Tool | Description |
|---|---|
| **gs_restore** | Tool for database-level import. It imports files exported using gs_dump. |

## 2.2 gs_loader for Importing Data

### Description

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.

**NOTE**

gs_loader does not support M-compatible databases.

gs_loader supports PDBs.

During a rolling upgrade from an earlier version to this version, do not use the gs_loader tool before all nodes are upgraded.

### Prerequisites

- Before using gs_loader, ensure that the gs_loader version is consistent with the gsql version and database version.
- After installing and deploying the gs_loader service, 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.
- 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
```

### 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 tool package to the created directory.

Taking the tool package for EulerOS as an example, upload the gsql tool package **GaussDB-Kernel_***Database version number_OS version number***_64bit_gsql.tar.gz** in the software installation package to the created directory.

**Step 3** Go to the directory where the tool package is located 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 number corresponds to the gsql tool version number. You can directly query the gsql client version number to verify the client version information.

```
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**

## Permission

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

The GUC parameter **enable_copy_error_log** controls whether to use the error table gs_copy_error_log and log table gs_copy_summary_log, and is set to **off** by default, indicating that the error table and log table are not used. Error records are directly recorded in the .bad file of gs_loader, and logs are directly recorded in the log file of gs_loader. If this parameter is set to **on**, the error table gs_copy_error_log and log table gs_copy_summary_log are used to insert error records into the error table. For details about the error table, see **Table 1 Error table gs_copy_error_log**. For details about the log table, see **Table 2-3**.

```
-- Enable separation of duties.
gs_guc set  -Z datanode -N all -I all -c "enableSeparationOfDuty = on"

-- Disable separation of duties.
gs_guc set  -Z datanode -N all -I all -c "enableSeparationOfDuty = off"

-- When enable_copy_error_log is set to on, the error table gs_copy_error_log is used.
gs_guc reload -Z datanode -N all -I all -c "enable_copy_error_log = on"
```

**Table 2-2** Error table pg_catalog.gs_copy_error_log

| Column | Type | Description |
|---|---|---|
| relname | text | Table name in the form of *Schema name*.*Table name*. |
| begintime | timestamp with time zone | Time when a data format error was reported. |
| filename | text | Name of the source data file where a data format error occurs. |

| Column | Type | Description |
|---|---|---|
| lineno | bigint | Number of the row where a data format error occurs in a source data file. |
| rawrecord | text | Raw record of a data format error in the source data file. |
| detail | text | Error details. |
| custom_id | text | Value of **copy_custom_id** in a copy. |

**Table 2-3** Columns in the pg_catalog.gs_copy_summary_log table

| 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 import transaction. |
| 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. |

| Column | Description |
|---|---|
| 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. |
| custom_id | Value of **copy_custom_id** in a copy. |

If **enableSeparationOfDuty** is set to **off**, the user can be a common database user or an administrator. If the user is a common user, administrators need to grant permissions to the common user. An administrator can use it directly.

1. Create a user as an administrator.
   ```
   gaussdb=# CREATE USER load_user WITH PASSWORD '********';
   ```
2. (Optional) Grant permissions on the pg_catalog.gs_copy_summary_log table to the new user.

   NOTE

   - The public.gs_copy_summary table used by gs_loader is changed to pg_catalog.gs_copy_summary_log. The original public.gs_copy_summary table is discarded. In addition, the system view pg_catalog.gs_copy_summary is added to query data in pg_catalog.gs_copy_summary_log for query compatibility. The name of the public.gs_copy_summary table may be the same as that of a user table. Therefore, exercise caution when migrating data to the pg_catalog.gs_copy_summary_log table and make sure the table is not a user table. After data migration is complete, delete public.gs_copy_summary. Because pg_catalog.gs_copy_summary exists, running the **\d(+)** gsql meta-command will preferentially find pg_catalog.gs_copy_summary, though public.gs_copy_summary is not displayed in the list because it is a system view. You can directly reference public.gs_copy_summary.
   - The pg_catalog.gs_copy_summary_log table is compatible only with the SELECT operation and is incompatible with the DELETE and TRUNCATE operations.

   ```
   gaussdb=# GRANT INSERT,SELECT,DELETE ON  pg_catalog.gs_copy_summary_log To load_user;
   ```
3. (Optional) Grant permissions on the error table gs_copy_error_log to the new user.

   NOTE

   - The public.pgxc_copy_error_log table used by gs_loader is changed to pg_catalog.gs_copy_error_log. The original public.pgxc_copy_error_log table is discarded. In addition, the system view pg_catalog.pgxc_copy_error_log is added to query data in pg_catalog.gs_copy_error_log for query compatibility. The name of the public.pgxc_copy_error_log table may be the same as that of a user table. Therefore, exercise caution when migrating data to the pg_catalog.gs_copy_error_log table and make sure the table is not a user table. After data migration is complete, delete public.pgxc_copy_error_log. Because pg_catalog.pgxc_copy_error_log exists, running the **\d(+)** gsql meta-command will preferentially find pg_catalog.pgxc_copy_error_log, though it is not displayed in the list because it is a system view. You can directly reference public.pgxc_copy_error_log.
   - The pg_catalog.gs_copy_error_log table is compatible only with the SELECT operation and is incompatible with the DELETE and TRUNCATE operations.

```
gaussdb=# GRANT INSERT,SELECT,DELETE ON  pg_catalog.gs_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.

**NOTE**

The pgxc_copy_error_log and gs_copy_summary tables always exist in the pg_catalog schema. In the setting of **search_path**, the pg_catalog schema takes precedence over user schemas. Therefore, to search for tables in a user schema by running the **\d(+)** gsql meta-command in the separation of duties scenario, explicitly specify the user schema.

1. Create a user as the initial user.
   ```
   gaussdb=# CREATE USER load_user WITH PASSWORD '********';
   ```
2. Switch to the new user as the initial user.
   ```
   gaussdb=# \c - load_user
   ```
3. (Optional) Create a gs_copy_summary table and add an index.

   **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 log table and do not need to create it. Otherwise, you need to create the log table.

   ```
   gaussdb=# 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);
   gaussdb=# CREATE INDEX gs_copy_summary_idx ON load_user.gs_copy_summary(id);
   ```
4. (Optional) Create a pgxc_copy_error_log table and add an index.

   **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 the pgxc_copy_error_log table.
   gaussdb=# CREATE TABLE load_user.pgxc_copy_error_log (relname varchar, begintime timestamptz,
   filename varchar, lineno int8, rawrecord text, detail text);

   -- Create an index.
   gaussdb=# CREATE INDEX copy_error_log_relname_idx ON load_user.pgxc_copy_error_log(relname);
   ```

## Format

```
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)]
[, ...]
)]
```

## Parameters

**Table 2-4** gs_loader parameters

| Parameter | Description | Parameter Type/Value Range |
|---|---|---|
| help | Displays help information. | - |
| user | Database connection user (equivalent to **-U**). | String |
| -U | Database connection user (equivalent to **user**). | String |
| passwd | User password (equivalent to **-W**). | String |
| -W | User password (equivalent to **passwd**). | String |
| db | Database name. This parameter is 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**.<br>If multiple host addresses are specified, the primary node is connected by default. | See **Table 1-9**. |
| -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**.<br>If multiple host addresses are specified, the primary node is connected by default. | See **Table 1-9**. |

| Parameter | Description | Parameter Type/Value Range |
|---|---|---|
| port | 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 **Table 1-9**. |
| -p | 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 **Table 1-9**. |
| create | Specifies whether to create the pgxc_copy_error_log and gs_copy_summary tables. In the current version, the two tables are created by default. Therefore, this parameter is meaningless. This parameter is reserved only for compatibility. | 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 |

| Parameter | Description | Parameter Type/Value Range |
|---|---|---|
| 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**. |
| limit | Specifies the maximum number of rows that can be imported. | Integer. By default, the value is infinite. |
| bindsize | Only syntax compatibility is implemented, but functions are not implemented. | - |
| rows | Number of rows of data to be imported before a commit. | The value is an integer, in the range [1,2147483647]. |
| compatible_nul | Specifies whether to enable the compatibility of NUL character (0x00) in data. After this function is enabled, if NUL characters exist in a data file, the NUL characters are converted to space characters (0x20), and then the data file is processed and imported. | The value can be **true** or **false**. The default value is **true**. |
| compatible_illegal_chars | Specifies whether to enable error tolerance for invalid characters. The error tolerance rules and restrictions are the same as those of **COMPATIBLE_ILLEGAL_CHARS** in the COPY syntax. For details, see the description about the **COMPATIBLE_ILLEGAL_CHARS** parameter in **COPY_OPTION** in "SQL Syntax > COPY" in *Developer Guide*. | The value can be **true** or **false**. The default value is **false**. |

| Parameter | Description | Parameter Type/Value Range |
|---|---|---|
| parallel | Specifies the degree of parallelism for data import. If the degree of parallelism is greater than 1, parallel import is enabled. If the degree of parallelism is 1, data is imported in serial mode. The maximum degree of parallelism cannot exceed twice the number of CPU cores on the client. When the client runs in a container, the number of CPUs obtained is the number of CPUs on the host, which may be greater than the number of CPUs that can be used by the container. You are advised to set the degree of parallelism to a value less than twice the number of CPUs that can be used by the client. In addition, this capability is implemented by multiple concurrent transactions through multiple threads. The actual degree of parallelism is limited by the thread pool model of the server and a high degree of parallelism will increase the pressure on the server. Set the degree of parallelism based on the actual situation. | The value is an integer ranging from 1 to twice the number of CPU cores. The default value is **1**. |
| binary | Specifies whether the binary file is exported in COPY binary mode. | The value can be **true** or **false**. The default value is **false**. |

NOTICE

- 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.
- OPTIONS, INFILE, and BADFILE are not supported. Syntax errors are not reported only in specific scenarios.
- gs_loader uses a .bad file to record errors from the **rawrecord** column in an error table if the GUC parameter **enable_copy_error_log** is set to enable the error table. If a raw record cannot be read due to unrecognized coding, the record will be marked as a blank line in the .bad file.
- 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.
- 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.
- Frequent commit of a small amount of data affects the data import performance. You are advised to set the **rows** parameter properly to ensure that the amount of data committed each time is greater than 5 MB. For common 16 vCPU 128 GB servers, in the scenario where one primary node and two standby nodes are deployed and 13 GB data is imported to a table with five columns. The rate of multiple commits is about 10 MB/s, which is basically the same as that of a single commit (5 MB data is committed each time).
- The **compatible_nul** parameter controls the value of the GUC parameter **loader_support_nul_character**.
  - **compatible_nul=true** corresponds to session-level **set loader_support_nul_character='s2'**.
  - **compatible_nul=false** corresponds to session-level **set loader_support_nul_character='s1'**.

    You are advised to set this parameter using the CLI. The priority of setting this parameter using **compatible_nul** is higher than that of GUC parameter setting.
- Currently, gs_loader supports compatibility only when data files contain NUL characters. It does not support NUL characters in .ctl control files. If the .ctl file contains the NUL character, unexpected problems may occur.
- When gs_loader is used to import data, if transcoding is not required, the maximum size of a single row of data is 1 GB minus 1 byte. If transcoding is required, the maximum size of a single row of data is 256 MB minus 1 byte. If the data volume in a single row is too large and the value of **max_process_memory** is too small, an error message is displayed, indicating that the memory is insufficient. In this case, you need to adjust the value of **max_process_memory** and try again.

- It is recommended that the size of a single file to be imported be less than or equal to 1 GB. gs_loader has no limit on the size of a single file to be imported. However, importing a large file is time-consuming. Therefore, you are advised to split a large data file, start multiple gs_loader processes to append data to the table. (If the truncate operation is required, run the **truncate** command separately instead of writing the truncate operation to the control file.) When the CPU resources are sufficient, this method can effectively improve the import speed.
- In a basic data migration scenario, you are advised to delete the index on the table and disable the trigger on the table. After the data migration is complete, rebuild the index and restore the trigger on the table. This is helpful to improve the import performance.
- After the **binary** parameter is set to **true**, the following requirements must be met:
  - The data file must be a binary file exported in \COPY binary mode. However, the data file exported in this mode has poor compatibility and portability. You are advised to use \COPY to import the data file.
  - gs_loader converts the syntax in the control file to the simplest syntax in \COPY binary mode, that is, \COPY *table_name* FROM *'binary_file_path'* BINARY;. Only the import mode, table name, as well as control, data, binary, guc_param, and database connection parameters in the control file are parsed. Other parameters are not parsed and do not take effect.
  - The command lines and control files of gs_loader must meet the following requirements:
    - Character set configuration is not supported.
    - The WHEN filter and DISCARD operation are not supported.
    - If **enable_copy_error_log** is set to **off**, error data cannot be directly written into .bad files, and log data cannot be directly written into log files. If **enable_copy_error_log** is set to **on**, error data cannot be directly written into the error table, and log data cannot be directly written into the log table.
    - The CSV mode is not supported, delimiters and wrappers cannot be specified, and the TRAILING NULLCOLS syntax is not supported.
    - Data type configuration, POSITION configuration, and column expression usage are not supported.
    - The **FILLER**, **CONSTANT**, **SEQUENCE**, and **NULLIF** parameters are not supported.
    - The **skip**, **rows**, **compatible_nul**, **compatible_illegal_chars**, and **parallel** parameters are not supported.
- If **parallel** is set to a value greater than **1**:
  - If **binary** is set to **true**, the setting of **parallel** becomes invalid and data is imported in serial mode.
  - **OPTIONALLY ENCLOSED BY** or **FIELDS CSV** cannot be set in the control file.
  - The SEQUENCE column cannot be set in the control file.
  - Data may not be imported in the sequence specified in the data file. If a table contains auto-increment columns, the sequence of values in the

auto-increment columns may be different from that in the data file after the table is imported.

- If the **errors** parameter is also used, the **errors** parameter indicates the maximum number of error lines allowed for each subtask.
- If the **skip** parameter is also used, the **skip** parameter indicates the number of rows to be skipped at the beginning of the entire data file.
- If the **rows** parameter is also used, the subtasks committed each batch are calculated independently.
- When the CPU and memory resources of the client and the CPU and memory resources, idle threads, and network bandwidth of the server do not have bottlenecks and the total proportion of bad and discarded files does not exceed 1%, the performance is improved by at least 1.5, 3, or 5 times compared with serial import when the degree of parallelism is **2**, **4**, or **8**.
- Each time the degree of parallelism increases by 1, the required client memory increases by about 10 MB and the required server memory increases by about 35 MB.

- If the GUC parameter **support_zero_character** is set to **on**, character 0x00 can be written to and read from the database. When gs_loader imports data, 0x00 is imported in the original style instead of being converted to 0x20 due to other compatibility parameters. Regarding the processing of 0x00, the parameter priority is as follows: **support_zero_character** > **copy_special_character_version** > **compatible_illegal_chars** > **loader_support_nul_character**.
- When **copy_special_character_version** is set to **'no_error'**, the setting of **copy_special_character_version** is prior to that of **compatible_illegal_chars**. Invalid characters will be imported without conversion.
- You are advised to use digits, letters, Chinese characters, and file path separators ('/') for parameter values; other characters may lead to shell command parsing anomalies.

---

- CHARACTERSET

  Character set.

  Value range: a string.

  Note: 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 CLI, 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 ("").

  NOTE

  - In the current CSV mode, newlines that are enclosed with double quotation marks are considered as part of the column data.
  - In CSV mode, if GUC parameter **a_format_copy_version** is set to **'s1'**, spaces at the beginning of column values are skipped. If the first non-space character in a column is not an enclosed character, the enclosure setting is ignored. If no enclosed character is matched and the end of the line is matched first, an error is reported.
  - In CSV mode, if GUC parameter **support_zero_characters** is disabled and **compatible_nul** or **compatible_illegal_chars** is used to be compatible with the 0x00 character, the 0x00 character at the beginning of column values is processed as 0x20 and deleted because the conversion from 0x00 to 0x20 occurs before the spaces at the beginning of column values are skipped.

- 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. The NUL character cannot be set as a separator.

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

NOTE

After enabling NUL character compatibility (**compatible_nul** set to **true**), if the specified separator is a space character (0x20), note that the separator is the space character that exists in the data file instead of the space character converted from the NUL character.

- 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.

  NOTE

  - When you set **OPTIONALLY ENCLOSED BY { 'string' }**, there should be no quotation mark on the left of the data; otherwise, the number of quotation marks on either left or right must be an odd number but they 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.

  NOTE

  - When the GUC parameter **enable_copy_when_filler** is set to **on** (default value), data can be filtered based on the **FILLER** column. When the GUC parameter **enable_copy_when_filler** is set to **off**, this usage is not supported.
  - The WHEN condition cannot be followed by special characters such as '\0' and '\r'.

- POSITION ({ start:end })

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

  NOTICE

  If **POSITION** is specified to use a fixed formatter in the \COPY statement after conversion, the newline characters in columns cannot be properly processed.

- "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.

  **NOTE**

  Currently, FILLER and POSITION ({ *start:end* }) cannot be used at the same time.

- 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

  Returns **NULL** if **a_format_copy_version** is set to **'s1'** and the data in the specified columns contains only whitespace characters; otherwise, it returns **trim (COL)**, which is equivalent to the column expression "nullif(trim(COL), '')".

  Sets columns without **NULLIF** specified to null if **a_format_copy_version** is not set to **'s1'** in multi-row import scenarios and no sysdate, constant, position, or column expression is specified after the column names.

  Currently, only the COL POSITION() CHAR NULLIF (COL=BLANKS) syntax is supported. For details, see **Example 2** and **Example 3**.

  **NOTE**

- 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. For details, see **Example 4** and **Example 5**.

- 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.
- The default length, POSITION, and delimiter are read from the current position to the line terminator.
- 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.

- DATEA:

  Reads data based on a column separator and converts the value to the DATEA 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. For details, see **Example 6**.

  - RAW:

    Parses each character into an ASCII code value. The backslash (\) is not used as an escape character. For details, see **Example 7**.

    Restriction: Separators cannot be used in RAW data.

NOTE

- 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, the SYSDATE and CONSTANT operations cannot be used together with the POSITION operation.
- 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**.
- When POSITION is specified, spaces at the end of a column are not omitted when the column content is processed. If POSITION is not specified, spaces at the end of the column content will be omitted. To retain spaces, ensure that **a_format_version** has been set and add **set behavior_compat_options='char_coerce_compat';** to the file specified by **guc_param**. For details, see "behavior_compat_options" in the *Administrator Guide*.
- 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:

  ```
  11,22,33,
  ```

  The import result is as follows:

  ```
  loader=# select * from gsloader;
  id |  text  |     gmt_create      | create_str
  ```

```
----+--------+---------------------+------------
11 | 2023-02-08 16:00:54 | 22 |  33
```

## Examples

Example 1

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

1. Create a user (as an administrator).
   ```
   gaussdb=# CREATE USER load_user WITH PASSWORD '********';
   ```
2. (Optional) Grant permission on the gs_copy_summary_log table 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 log table and do not need to create it. Otherwise, you need to create the log table.
   ```
   gaussdb=# GRANT INSERT,SELECT,DELETE ON  pg_catalog.gs_copy_summary_log To load_user;
   ```
3. (Optional) Grant permission on the gs_copy_error_log table 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.
   ```
   gaussdb=# GRANT INSERT,SELECT,DELETE ON  pg_catalog.gs_copy_error_log To load_user;
   ```
4. Switch to another user.
   ```
   gaussdb=# \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).
   ```
   gaussdb=# CREATE USER load_user WITH PASSWORD '********';
   ```
2. Switch to the **load_user** user (as the initial user).
   ```
   gaussdb=# \c - load_user
   ```
3. (Optional) Create a gs_copy_summary table and add an index.

   **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 log table and do not need to create it. Otherwise, you need to create the log table.
   ```
   gaussdb=# 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);
   gaussdb=# CREATE INDEX gs_copy_summary_idx ON load_user.gs_copy_summary(id);
   ```
4. (Optional) Create a pgxc_copy_error_log table and add an index.

   **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.
   ```
   gaussdb=# CREATE TABLE load_user.pgxc_copy_error_log (relname varchar, begintime timestamptz,
   filename varchar, lineno int8, rawrecord text, detail text);
   gaussdb=# 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.

```
gaussdb=# 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.

(On the gs_loader client) Before importing data, ensure that the gs_loader tool has the execute permission on the gs_loader tool. 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 guc_param=guc.txt 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**

**NOTE**

gs_copy_summary is used to record the called COPY syntax and its details.
*[badfile]*_**bad.log** is used to record error data and its 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. If the error table is used 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 error and log tables.

Example 2

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.)

```
-- For the file corresponding to the loader, search for the first row of the data text to find the source data.
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)
```

Example 3

NULLIF use cases:

```
-- Create the gsloader_test_nullif table.
gaussdb=# 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)
);

-- View the 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

-- View the 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

-- View the import result. The import is successful.
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.

Example 4

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.
gaussdb=# create table t_test(id int, text varchar(5));

-- View the data file test.csv.
addf2,bbbbaaa,20220907,

-- View the 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')"
)

-- Use guc_param to set the a_format_copy_version parameter.
cat test_guc.txt
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

-- View the import result. The import is successful.
gaussdb=# select * from t_test;
 id | text
----+-------
  5 | aaaaa
(1 row)
```

Example 5

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.
gaussdb=# create table test(mes int, mes1 text, mes2 float8, mes3 timestamp with time zone, mes4
INTEGER);

-- View the 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

-- View the 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
)

-- Use guc_param to set the a_format_copy_version parameter.
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

-- View the import result. The import is successful.
gaussdb=# 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
```

Example 6

```
-- Create a table.
gaussdb=# create table t_spec(col1 varchar(10), col2 varchar(10));

-- View the data file.
cat t_spec.txt
1234,5678,

-- View the 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
)

-- Use guc_param to set the a_format_copy_version parameter.
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

-- View the import result. The import is successful.
gaussdb=# select * from t_spec;
   col1    | col2
-----------+-------
 741618482 | 13612
(1 row)
```

Example 7

```
-- Create a table.
gaussdb=# create table t_raw(col raw(50));

-- View the data file.
cat t_raw.txt
12\n\x78!<~?'k^(%s)>/c[$50]

-- View the control file.
cat t_raw.ctl
Load Data
TRUNCATE INTO TABLE t_raw
TRAILING NULLCOLS(
col position(1:50) raw
)

-- Use guc_param to set the a_format_copy_version parameter.
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

-- View the import result. The import is successful.
gaussdb=# select * from t_raw;
                        col
--------------------------------------------------------
 31325C6E5C783738213C7E3F276B5E282573293E2F635B2435305D
(1 row)
```

## 2.3 gs_dump for Exporting Database Information

### Description

gs_dump, provided by GaussDB, is used to export database information. You can export a database or its objects (such as schemas, tables, and views). The database can be the default **postgres** database or a user-specified database.

- gs_dump supports SSL encrypted communication.
- gs_dump can be used to export PDB data. When exporting a PDB, you can export a PDB or its objects (such as schemas, tables, and views).
- When gs_dump is used to export data, other users can still access (read or write) the database.
- gs_dump can export complete, consistent data. For example, if gs_dump is started to export database A at T1, data of the database at that time point will be exported, and modifications on the database after that time point will not be exported.
- gs_dump supports the export of text files that are compatible with the V1 database.
- gs_dump can export database information to a plain-text SQL script file or archive file.
  - Plain-text SQL script: It contains the SQL statements required to restore the database. You can use **gsql for Connecting to a Database** to execute the SQL script. With only a little modification, the SQL script can rebuild a database on other hosts or database products.
  - Archive file: It contains data required to restore the database. It can be a tar-, directory-, or custom-format archive. For details, see **Table 2-5**.

- gs_dump supports four export file formats, which are specified by **[-F or --format=]**. For details, see **Table 2-5**.

**Table 2-5** Formats of exported files

| Format | Value of -F | Description | Suggestion | Import Tool |
|---|---|---|---|---|
| Plain-text | p | A plain-text script file containing SQL statements and commands. The commands can be executed on gsql, a command line terminal, to rebuild database objects and load table data. | For a small-sized database or the exported SQL file needs to be modified, the plain-text format is recommended. | Before using **gsql for Connecting to a Database** to restore database objects, you can use a text editor to edit the plain-text export file as required. |
| Custom | c | A binary file that allows the restoration of all or selected database objects from an exported file. | You are advised to use custom-format archive files for medium or large database. | You can use **gs_restore** to import database objects from a custom-, directory-, or tar-format archive. |
| Directory | d | A directory containing directory files of database objects and the data files of tables and BLOBs. | Database objects and data files need to be stored and exported in different directories. Therefore, the directory format is recommended. | |
| .tar archive | t | A tar-format archive that allows the restoration of all or selected database objects from an exported file. The tar file cannot be further compressed and has an 8-GB limitation on the size of each single file. | You need to export the archive result and pack it. The tar format is recommended. | |

NOTE

- To reduce the size of an exported file, you can use the gs_dump tool to compress it to a directory archive file or custom-format file. When a directory archive or custom-format archive is generated, medium compression is applied by default. Archived exported files cannot be compressed using gs_dump.
- In M-compatible mode, do not import or export data between instances with different **lower_case_table_names** parameter settings. Otherwise, data may be lost.

## Precautions

- Before using gs_dump, ensure that the gs_dump version is consistent with the database version. gs_dump of a later version may not be fully compatible with kernel data of an earlier version.
- gs_dump is not suitable for scenarios where there are too many objects (such as tables, views, and indexes) in the database. If the number of objects in the database exceeds 100,000 or the dependency between objects is too complex, the export using gs_dump takes a long time.
- The generated columns are not dumped when gs_dump is used.
- When gs_dump is used, the IMCV metadata (gs_imcv system catalog) created in the HTAP table is not dumped.
- Do not modify the files and contents exported using the **-F c/d/t** formats. Otherwise, the restoration may fail. For files exported using the **-F p** format, you can edit the exported files with caution if necessary.
- To ensure the data consistency and integrity, gs_dump acquires a share lock on a table to be dumped. If a share lock has been set for the table in other transactions, gs_dump locks the table after it is released. If the table cannot be locked within the specified time, the dump fails. You can customize the timeout interval to wait for lock release by specifying the **--lock-wait-timeout** parameter.
- Stored procedures and functions cannot be exported in encrypted mode.
- For materialized views, this tool supports only definition export. After importing materialized views, you need to manually run the REFRESH statement to restore data.
- For temporary objects, this tool can export only global temporary tables.
- This tool cannot be used on standby nodes.
- When gs_dump is used to export partitioned indexes, the attributes of some index partitions cannot be exported, for example, the unusable status of index partitions. You can query the PG_PARTITION system catalog or ADM_IND_PARTITIONS or ADM_IND_SUBPARTITIONS view to obtain the attributes of an index partition. You can run the **ALTER INDEX** statement to manually set the attributes of an index partition.
- For scheduled tasks, this tool can export only scheduled tasks created using CREATE EVENT from a B-compatible database. Scheduled tasks created using advanced packages cannot be exported in this case. Only the initial user can export all scheduled tasks. Users with the SYSADMIN permission can export scheduled tasks owned by themselves. Common users without related permissions cannot export scheduled tasks.
- gs_dump does not export user-defined tokenweight dictionaries. You can manually create the corresponding tokenweight dictionary according to the

dictionary information displayed in the error message "WARNING: dictionary xx cannot be automatically exported, please create it manually."

- In the multi-tenancy scenario, when gs_dump is used to export data, the template PDB cannot be exported, and closed PDBs cannot be exported.
- In the multi-tenancy scenarios, when a common user uses gs_dump to export data, only the database objects and data on which the user has permission can be exported.
- If the multi-tenancy function is disabled, gs_dump cannot export PDBs and objects in PDBs.
- If a table created by the initial user exists in the database and the table contains expression indexes of UDFs, after using gs_dump to export the table, system administrators must import the table as the initial user when using gsql or gs_restore to import the table. Otherwise, the indexes fail to be created due to security reasons.
- Common users cannot export directories and synonyms. If a common user attempts to export related data, the message "WARNING: xx not dumped because current user is not a superuser" is displayed.
- If a scheduled task created using the CREATE EVENT syntax cannot be exported, check whether **job_style** of the scheduled task in the gs_job_attribute table is **EVENT**. If not, create the scheduled task again.
- If local data exists in **template1** of the source database instance, restore the output of gs_dump to an empty database (copy from **template0** instead of **template1**). Otherwise, an error may occur because the definition of the added object is copied. To create an empty database without any local additions, copy data from template0 rather than template1. For example:
  ```
  CREATE DATABASE foo WITH TEMPLATE template0;
  ```
- The size of a single tar file must be smaller than 8 GB. (This is the tar file format limitations.) The total size of a .tar archive and any of the other output formats are not limited, except possibly by the OS.
- The dump file generated by gs_dump does not contain the statistics used by the optimizer to make execution plans. Therefore, you are advised to run **ANALYZE** after restoring from a dump file to ensure optimal performance. The dump file does not contain the **ALTER DATABASE... SET** command, database users, and other installation settings. These settings are dumped by gs_dumpall.
- The current design of gs_dump is incompatible with separation of duties. In separation of duties, gs_dump can be used only by the initial user. The file to be imported must also be executed by the initial user.
- gs_dump does not export invalid functions. Before exporting functions, ensure that the functions are valid. If a function is invalid, an alarm is generated, indicating that the function corresponding to the OID is invalid.
- If the parameter type of a non-PACKAGE function is the table column type (table.column%type) or view column type (view.column%type) and the column type is a basic type, the parameter type is converted to the corresponding basic type in the exported function definition. Changing the column type of a table by executing **ALTER TABLE MODIFY COLUMN** and then running **ALTER FUNCTION COMPILE**, or changing the column type of a view query table, rebuilding the view, and executing **ALTER FUNCTION COMPILE** does not change the parameter type in the **PROARGSRC** column of the function in the PG_PROC system catalog. In the exported function

definition, the parameter type remains the basic type before the object column type is changed.

- For a view that depends on functions in a PACKAGE, deleting the PACKAGE will cause a cascading deletion of the view. When the **ddl_invalid_mode** parameter is set for cascading invalidation, deleting the PACKAGE will result in an error during view export.

## Format

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

## Parameters

- DBNAME

  Specifies the database to be connected. No short or long option is required. For example:

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

  Or,

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

- OPTION

  There are three types of parameters: general parameters, dump parameters, and connection parameters.

  Common parameters:

  – -f, --file=FILENAME

    Sends the output to the specified file or directory. If this parameter is omitted, the standard output is generated. If the output format is **-F c**, **-F d**, or **-F t**, the **-f** parameter must be specified. If the value of the **-f** parameter contains a directory, the current user must have the read and write permission on the directory.

  – -F, --format=c|d|t|p

    Selects an exported file format. The formats are as follows:

    ▪ **c|custom**: outputs a custom-format archive as a directory to be used as the input of gs_restore. This is the most flexible output format in which users can manually select it and reorder the archived items during the restoration process. An archive in this format is compressed by default.

    ▪ d|directory: creates a directory containing directory files and the data files of tables and BLOBs.

    ▪ **t|tar**: outputs a tar-format archive as the input of gs_restore. The .tar format is compatible with the directory format. Extracting a .tar archive generates a valid directory-format archive. However, the .tar archive cannot be further compressed and has an 8-GB limitation on the size of a single table. The order of table data items cannot be changed during restoration.

    ▪ p|plain: generates a text SQL script file. This is the default value.

  – -v, --verbose

Specifies the verbose mode. If it is specified, gs_dump writes detailed object comments and number of startups/stops to the dump file, and progress messages to standard errors.

– -V, --version

  Prints the gs_dump version and exits.

– -Z, --compress=0-9

  Specifies the used compression level.

  Value range: 0 to 9

  - **0** indicates no compression.
  - **1** indicates the lowest compression ratio and the fastest processing speed.
  - **9** indicates the highest compression ratio and the slowest processing speed.

  For the custom-format archive, this option specifies the compression level of a single table data segment. By default, data is compressed at a medium level. The .tar archive format and plain-text format do not support compression currently.

– --lock-wait-timeout=TIMEOUT

  Specifies the timeout period for waiting for a shared table lock when the dump starts. If no such lock is obtained in the specified period, the dump fails. The timeout interval can be specified in any of the formats accepted by SET statement_timeout.

– -?, --help

  Displays help information about gs_dump command line parameters.

Dump parameters:

– -a, --data-only

  Generates only the data, not the schema (data definition). Dumps the table data, LOBs, and sequence values.

– -b, --blobs

  Reserved for function extension. The option is not recommended.

– -c, --clean

  Before writing the command of creating database objects into the backup files, writes the command of cleaning (deleting) database objects to the backup files. If no objects exist in the target database, gsql or gs_restore probably displays some error information.

  This parameter is used only for the plain-text format. For the archive format, you can specify the option when using gs_restore.

– -C, --create

  The backup file content starts with the commands of creating the database and connecting to the created database. If the command script is executed in this mode, the database that is connected before the script is executed is not affected.

  This parameter is used only for the plain-text format. For the archive format, you can specify the option when using gs_restore.

**NOTE**

- In the multi-tenancy scenario, this option cannot be used when gs_dump is used to export a specified PDB.
- This option is not supported in M-compatible databases. You must create a database in M-compatible mode on the target instance, export data from the source instance, and connect to the newly created M-compatible database on the target instance to import data.

– -E, --encoding=ENCODING

Creates the dump in the specified character set encoding. By default, the dump file is created in the database encoding. Setting the environment variable *PGCLIENTENCODING* to the required dump encoding has the same function as this option.

**NOTE**

If transcoding is required for the specified dump encoding and data in the table contains invalid characters, the error message "invalid byte sequence" will be displayed during export. You are advised to specify the **-s** parameter when using gs_dump to export only definitions and execute **COPY** with the encoding fault tolerance option separately to export and import data.

– -n, --schema=SCHEMA

Dumps only schemas matching the schema names. This option contains the schema and all its contained objects. If this option is not specified, all non-system schemas in the target database will be dumped. Multiple **-n** options can be transferred to select multiple modes. The schema parameter is interpreted as a pattern according to the same rule used by the **\d** command of gsql. Therefore, multiple schemas can also be selected by writing wildcard characters in the pattern. When you use wildcard characters, quote patterns to prevent the shell from expanding the wildcard characters.

**NOTE**

- If **-n** is specified, gs_dump does not dump any other database objects that the selected schemas might depend upon. Therefore, there is no guarantee that the results of a specific-schema dump can be automatically restored to an empty database.
- If **-n** is specified, the non-schema objects are not dumped.
- In M-compatible mode, if a database with **templatem** is created by running **CREATE DATABASE**, data is exported by specifying **db_name**; if a database is created by running **CREATE DATABASE** *db_name*, data is exported by specifying **-n** because such database is equivalent to a schema.
- GaussDB automatically converts uppercase letters in object names to lowercase letters. If a schema name contains uppercase letters, add extra quotation marks, for example, **-n '"Sch1"'** or **-n "\"Sch1\""**.
- In M-compatible mode, the value of this parameter is affected by the GUC parameter **lower_case_table_names**. In case-sensitive mode (**lower_case_table_names** set to **0**), the value of this parameter must be case-sensitive. If the value contains uppercase letters, add extra quotation marks. Otherwise, the result is the same as that in lowercase. In case-insensitive mode (**lower_case_table_names** set to **1**), the value of this parameter must be in lowercase.

Multiple schemas can be dumped. For example, in the following example, both **sch1** and **sch2** are dumped.

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

– -N, --exclude-schema=SCHEMA

  Does not dump any tables matching the table pattern. The pattern is interpreted according to the same rule as for -n. **-N** can be specified multiple times to exclude schemas matching any of the specified patterns.

  When both -n and -N are specified, the schemas that match at least one -n option but no -N is dumped. If -N is specified and -n is not, the schemas matching -N are excluded from what is normally dumped.

  Dump allows you to exclude multiple schemas during dumping. For example, in the following example, **sch1** and **sch2** are excluded during the dump.

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

  **NOTE**

  GaussDB automatically converts uppercase letters in object names to lowercase letters. If a schema name contains uppercase letters, add extra quotation marks, for example, **-N '"Sch1"'** or **-N "\"Sch1\""**.

– -o, --oids

  Dumps OIDs as parts of the data in each table. Use this option if your application references the OID columns in some way (for example, in a FOREIGN KEY constraint). If the preceding situation does not occur, do not use this parameter.

– -O, --no-owner

  Does not output commands to set ownership of objects to match the original database. By default, gs_dump issues the ALTER OWNER or SET SESSION AUTHORIZATION command to set ownership of created database objects. These statements will fail when the script is running unless it is started by a system administrator (or the same user who owns all of the objects in the script). To make a script that can be used by any user for restoration and give the user ownership of all objects, specify **-O**.

  This parameter is used only for the plain-text format. For the archive format, you can specify the option when using gs_restore.

– -s, --schema-only

  Dumps only the object definition (schema) but not data.

– -S, --sysadmin=NAME

  Reserved for function extension. The option is not recommended.

– -t, --table=TABLE

  Specifies a list of tables, views, sequences, or foreign tables not to be dumped. You can use multiple -t parameters or wildcard characters to specify tables.

  When you use wildcard characters, quote patterns to prevent the shell from expanding the wildcard characters.

  The **-n** and **-N** options have no effect when **-t** is used, because tables selected by using **-t** are not affected the **-n** and **-N** options.

**NOTE**

- The number of **-t** parameters must be less than or equal to 100.
- If the number of **-t** parameters is greater than 100, you are advised to use the **--include-table-file** parameter to replace some **-t** parameters.
- If **-t** is specified, gs_dump does not dump any other database objects that the selected tables might depend upon. Therefore, there is no guarantee that the results of a specific-table dump can be automatically restored to an empty database.
- **-t** *tablename* only dumps visible tables in the default search path. **-t** **.tablename* dumps tablename tables in all the schemas of the dumped database. **-t** *schema.table* dumps tables in a specific schema.
- **-t** *tablename* does not export trigger information from a table.
- If the table name contains uppercase letters, you need to add **\"** to the table name when using the **-t** parameter to specify the export. To export the **"abC"** table, specify **-t \"abC\"**. To export the **schema."abC"** table, specify **-t schema.\"abC\"**.
- **-t ""** does not match any table.
- In M-compatible mode, the value of this parameter is affected by the GUC parameter **lower_case_table_names**. In case-sensitive mode (**lower_case_table_names** set to **0**), the value of this parameter must be case-sensitive. If the value contains uppercase letters, add extra quotation marks. Otherwise, the result is the same as that in lowercase. In case-insensitive mode (**lower_case_table_names** set to **1**), the value of this parameter must be in lowercase.

For example, in the following example, both **schema1.table1** and **schema2.table2** are dumped.

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

– --include-table-file=FILENAME

Specifies the table file to be dumped.

– -T, --exclude-table=TABLE

Specifies a list of table, view, sequence, or foreign table objects not to be dumped. You can use multiple **-T** parameters or wildcard characters to specify multiple lists.

When both **-t** and **-T** are specified, it will dump the objects in the **-t** list, but not those in the **-T** list.

For example, in the following example, **table1** and **table2** are excluded during the dump.

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

– --exclude-table-file=FILENAME

Specifies the table file not to be dumped.

**NOTE**

Same as **--include-table-file**, the content format of this parameter is as follows:

schema1.table1

schema2.table2

......

– -x, --no-acl

Prevents the dumping of access permissions (**GRANT**/**REVOKE** commands). Only ACL objects are affected. Privilege objects are not affected.

– -q, --target=VERSION

  Exports text files compatible with databases of other versions. Currently, parameters of V1 and V5 are supported. If other parameters are specified, no error is reported but the setting does not take effect. The **v1** value means to export GaussDB v5 data as a text file compatible with GaussDB v1. The **v5** value means to export GaussDB v5 data as a text file, reducing errors that may occur when data is imported to a GaussDB v5 database.

  When **v1** is specified, you are advised to use it along with parameters such as **--exclude-guc="enable_cluster_resize"**, **--exclude-function**, and **--exclude-with**. Otherwise, an error may be reported during data import to a GaussDB v1 database.

– -g, --exclude-guc

  Reserved for function extension. The option is not recommended.

– --exclude-function

  Specifies that functions and stored procedures are not exported.

– --exclude-with

  Specifies that the description such as **WITH(orientation=row, compression=on)** is not added to the end of the exported table definition.

– --binary-upgrade

  Reserved for function extension. The option is not recommended.

  **NOTE**

  M-compatible databases do not support this option.

– --binary-upgrade-usermap="USER1=USER2"

  Reserved for function extension. The option is not recommended.

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

  Exports data by running the INSERT command with explicit column names {INSERT INTO table (column, ...) VALUES ...}. This will cause a slow restoration. However, since this option generates an independent command for each row, an error in reloading a row causes only the loss of the row rather than the entire table content.

  **NOTE**

  M-compatible databases do not support this option.

– --disable-dollar-quoting

  Disables the use of dollar sign ($) for function bodies, and forces them to be quoted using the SQL standard string syntax.

– --include-alter-table

  Exports columns that are deleted from a table.

– --disable-triggers

  Reserved for function extension. The option is not recommended.

- --exclude-table-data=TABLE

  Does not dump table data that matches any patterns. The pattern is interpreted according to the same rule as for -t.

  **--exclude-table-data** can be entered more than once to exclude tables matching any of several patterns. When the user needs the specified table definition rather than data in the table, this option is helpful.

  To exclude data of all tables in the database, see **-s, --schema-only**.
- --inserts

  Dumps data when the **INSERT** command (rather than the **COPY** command) is delivered, resulting in slow recovery.

  However, since this option generates an independent command for each row, an error in reloading a row causes only the loss of the row rather than the entire table content. Note that the restoration may fail if you rearrange the column order. The **--column-inserts** option is unaffected against column order changes.

  **NOTE**

  M-compatible databases do not support this option.
- --no-security-labels

  Reserved for function extension. The option is not recommended.
- --no-tablespaces

  Does not select any tablespaces. All the objects will be created during the restoration process, no matter which tablespace is selected when using this option.

  This parameter is used only for the plain-text format. For the archive format, you can specify the option when using gs_restore.
- --no-unlogged-table-data

  Reserved for function extension. The option is not recommended.
- --non-lock-table

  Shields the table locking behavior during the export using gs_dump. This parameter is called only between software interfaces. You are advised not to call this parameter.

  **NOTE**

  When gs_dump is used to export a table, an access share lock is added to the table to be exported in the entire transaction by default. This lock blocks the execution of concurrent DDL statements on the table, ensuring that no blocking occurs due to the impact on concurrent DDL statements.
- --quote-all-identifiers

  Forcibly quotes all identifiers. In the migration to a later version, additional keywords may be introduced. This option is helpful.

  **NOTE**

  M-compatible databases do not support this option.
- --section=SECTION

  Dumps specified sections (pre-data, data, and post-data).

- --serializable-deferrable

  Uses a serializable transaction for the dump to ensure that the used snapshot is consistent with later database status. Perform this operation at a time point in the transaction flow, at which everything is normal. This ensures successful transaction and avoids serialization failures of other transactions, which requires serialization again.

  This option has no benefits to disaster recovery. During the upgrade of the original database, loading a database copy as a report or other shared read-only dump is helpful. The option does not exist, dump reveals a status which is different from the commit sequence status of any transaction.

  This option will make no difference if there are no active read/write transactions when gs_dump is started. If the read/write transactions are active, the dump start time will be delayed for an uncertain period.

- --use-set-session-authorization

  Specifies that the standard SQL **SET SESSION AUTHORIZATION** command rather than **ALTER OWNER** is generated to determine the object ownership. When the **SET SESSION AUTHORIZATION** command is used to dump data, if the object history in the dump file is incorrect, the data may not be restored correctly. A dump by running **SET SESSION AUTHORIZATION** requires the system administrator permission, whereas running **ALTER OWNER** requires a lower permission. However, the SET SESSION AUTHORIZATION statement supports ciphertext passwords. The script exported using this parameter may not be restored. Therefore, you are advised not to use this parameter to export the script.

  NOTE

  - Application scope of SET SESSION AUTHORIZATION:
    - A system administrator can switch to a common user using the SET SESSION AUTHORIZATION statement, but cannot switch to the initial user, SYSADMIN, OPRADMIN, MONADMIN, POLADMIN, or AUDITADMIN.
    - Other users cannot switch to another user using the SET SESSION AUTHORIZATION statement.
  - If the **--use-set-session-authorization** parameter is used by a system administrator to export data and the imported data contains objects of the initial user, "ERROR: permission denied to set session authorization" is reported when the system administrator switches to the initial user due to security permission verification and **SET SESSION AUTHORIZATION application scope** limitations. As a result, the owner of the objects is changed from the initial user to the system administrator role used during the import. In this scenario, you are advised to use the initial user to import and export data.

- --with-encryption=AES128

  Specifies that dumping data needs to be encrypted using AES-128.

- --with-key=KEY

  The AES-128 key rules are as follows:

  - Consists of 8 to 16 characters.

- Contains at least three of the following character types: uppercase characters, lowercase characters, digits, and special characters (limited to ~!@#$ %^&*()-_=+\|[{}];:,<.>/?).

NOTE

Stored procedures and functions cannot be exported in encrypted mode.

– --with-salt=RANDVALUES

  gs_dumpall uses this parameter to transfer a random value.

– --include-extensions

  Includes extensions in the dump.

NOTICE

The extended function is for internal use only. You are advised not to use it.

– --include-depend-objs

  Includes information about the objects that depend on the specified object in the backup result. This parameter takes effect only if the **-t** or **--include-table-file** parameter is specified.

– --exclude-self

  Excludes information about the specified object from the backup result. This parameter takes effect only if the **-t** or **--include-table-file** parameter is specified.

– --pipeline

  Uses a pipe to transmit the password. This parameter cannot be used on devices.

– --dont-overwrite-file

  The existing files in plain-text, .tar, and custom formats will be overwritten. This option is not applicable to the directory format.

  For example:

  Assume that the backup.sql file exists in the current directory. If you specify -f backup.sql in the input command, and the backup.sql file is generated in the current directory, the original file will be overwritten.

  If the backup file exists and **--dont-overwrite-file** is specified, an error will be reported with the message that the dump file exists.

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

NOTE

- **-s/--schema-only** and **-a/--data-only** do not coexist.
- **-c/--clean** and **-a/--data-only** do not coexist.
- **--inserts/--column-inserts** and **-o/--oids** do not coexist, because OIDS cannot be set using the INSERT statement.
- **--role** must be used with **--rolepassword**.
- **--binary-upgrade-usermap** must be used with **--binary-upgrade**.
- **--include-depend-objs**/**--exclude-self** takes effect only when **-t**/**--include-table-file** is specified.
- **--exclude-self** must be used with **--include-depend-objs**.
- **--with-encryption=AES-128** supports only **-F p/plain**.
- **--with-key=KEY** supports only **-F p/plain**.
- **--with-salt=RANDVALUES** is called by gs_dumpall and does not require manual input.

Connection parameters:

– -h, --host=HOSTNAME

  Specifies the host name. If the value begins with a slash, it is used as the directory for the UDS. By default, the value is obtained from the *PGHOST* environment variable (if set).

  This parameter is used only for defining names of the hosts outside a database. For the localhost in the database, you can use IPv4 address **127.0.0.1** or IPv6 address **::1**.

– -p, --port=PORT

  Specifies the host port number. If the thread pool is enabled, you are advised to use the pooler port, that is, the host port number plus 1. By default, the value is obtained from the *PGPORT* environment variable (if set).

– -U, --username=NAME

  Specifies the username for connecting to a host. The initial user cannot be used for cross-node execution. By default, the value is obtained from the *PGUSER* environment variable (if set).

– -w, --no-password

  Never issues a password prompt. The connection attempt fails if the host requires password for authentication and the password is not provided in other ways. This parameter is useful in batch jobs and scripts in which no user password is required.

– -W, --password=PASSWORD

  Specifies the user password for connection. If the host uses the trust authentication policy, administrators do not need to enter the **-W** option. If the **-W** option is not provided and you are not a system administrator, the system will ask you to enter a password. To ensure system security, you are advised to enter the password in interactive mode.

– --role=ROLENAME

  Specifies a role name to be used for creating the dump. If this option is selected, the SET ROLE command will be issued after the database is connected to gs_dump. It is useful when the authenticated user (specified by **-U**) lacks the permissions required by gs_dump. It allows the user to switch to a role with the required permissions. Some installations have a policy against logging in directly as a system administrator. This option allows dumping data without violating the policy.

– --rolepassword=ROLEPASSWORD

  Password of the role

## Examples

In the following examples, **backup/MPPDB_backup.sql** indicates an exported file, where **backup** indicates the relative path of the current directory. **37300** indicates the port ID of the database server. **testdb** indicates the name of the database to be accessed.

NOTE

Before exporting files, ensure that the directory exists and you have the read and write permissions on the directory.

**Example 1**

Use gs_dump to export the full information of the **testdb** database. The exported **MPPDB_backup.sql** file is in plain-text format.

```
gs_dump -U omm -f backup/MPPDB_backup.sql -p 37300 testdb -F p
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-27 09:49:17]: The total objects number is 
356.
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-27 09:49:17]: [100.00%] 356 objects have 
been dumped.
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-27 09:49:17]: dump database testdb 
successfully
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-27 09:49:17]: total time: 1274  ms
```

**Example 2**

Use gs_dump to export the full information of the **testdb** database. The exported **MPPDB_backup.tar** file is in .tar format.

```
gs_dump -U omm -f backup/MPPDB_backup.tar -p 37300 testdb -F t
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-27 10:02:24]: The total objects number is 
1369.
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-27 10:02:53]: [100.00%] 1369 objects have 
been dumped.
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-27 10:02:53]: dump database testdb 
successfully
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-27 10:02:53]: total time: 50086  ms
```

**Example 3**

Use gs_dump to export the full information of the **testdb** database. The exported **MPPDB_backup.dmp** file is in custom format.

```
gs_dump -U omm -f backup/MPPDB_backup.dmp -p 37300 testdb -F c
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-27 10:05:40]: The total objects number is 
1369.
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-27 10:06:03]: [100.00%] 1369 objects have 
been dumped.
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-27 10:06:03]: dump database testdb 
successfully
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-27 10:06:03]: total time: 36620  ms
```

**Example 4**

Use gs_dump to export the full information of the **testdb** database. The exported **MPPDB_backup** file is in directory format.

```
gs_dump -U omm -f backup/MPPDB_backup -p 37300  testdb -F d
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-27 10:16:04]: The total objects number is 
1369.
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-27 10:16:23]: [100.00%] 1369 objects have 
been dumped.
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-27 10:16:23]: dump database testdb 
successfully
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-27 10:16:23]: total time: 33977  ms
```

**Example 5**

Use gs_dump to export the information of the **testdb** database, excluding the information of the table specified in the **/home/MPPDB_temp.sql** file. The exported MPPDB_backup.sql file is in plain-text format.

```
gs_dump -U omm -p 37300 testdb --exclude-table-file=/home/MPPDB_temp.sql -f backup/
MPPDB_backup.sql
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-27 10:37:01]: The total objects number is 
1367.
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-27 10:37:22]: [100.00%] 1367 objects have 
been dumped.
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-27 10:37:22]: dump database testdb 
successfully
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-27 10:37:22]: total time: 37017  ms
```

**Example 6**

Use gs_dump to export only the information about the views that depend on the **testtable** table.

```
gs_dump -U omm -s -p 37300 testdb -t PUBLIC.testtable --include-depend-objs --exclude-self -f backup/
MPPDB_backup.sql -F p
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-15 14:12:54]: The total objects number is 
331.
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-15 14:12:54]: [100.00%] 331 objects have 
been dumped.
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-15 14:12:54]: dump database testdb 
successfully
gs_dump[user='omm'][localhost][port='37300'][testdb][2018-06-15 14:12:54]: total time: 327  ms
```

**Example 7**

In the multi-tenancy scenario, use gs_dump to export the full information about the PDB named **testpdb**. The exported **backup_pdb.sql** file is in plain-text format.

```
gs_dump -U omm testpdb -f backup/backup_pdb.sql -p 20000 -F p
gs_dump[user='omm'][localhost][port='20000'][testpdb][2024-04-26 17:41:21]: The total objects number is 
459.
gs_dump[user='omm'][localhost][port='20000'][testpdb][2024-04-26 17:41:21]: [100.00%] 459 objects have 
been dumped.
gs_dump[user='omm'][localhost][port='20000'][testpdb][2024-04-26 17:41:21]: dump database testpdb 
successfully
gs_dump[user='omm'][localhost][port='20000'][testpdb][2024-04-26 17:41:21]: total time: 5427  ms
```

**Example 8**

In the multi-tenancy scenario, use gs_dump to export the full information about the PDB named **testpdb**. The exported **backup_pdb_t.tar** file is in tar format.

```
gs_dump -U omm testpdb -p 20000 -f backup/backup_pdb_t.tar -F t
gs_dump[user='omm'][localhost][port='20000'][testpdb][2024-04-26 17:02:40]: The total objects number is 
459.
gs_dump[user='omm'][localhost][port='20000'][testpdb][2024-04-26 17:02:40]: [100.00%] 459 objects have 
been dumped.
```

```
gs_dump[user='omm'][localhost][port='20000'][testpdb][2024-04-26 17:02:40]: dump database testpdb
successfully
gs_dump[user='omm'][localhost][port='20000'][testpdb][2024-04-26 17:02:40]: total time: 5506  ms
```

**Example 9**

In the multi-tenancy scenario, use gs_dump to export the full information about the PDB named **testpdb**. The exported **backup_pdb_c** file is in customized archive format.

```
gs_dump -U omm testpdb -p 20000 -f backup/backup_pdb_c -F c
gs_dump[user='omm'][localhost][port='20000'][testpdb][2024-04-26 16:57:19]: The total objects number is
459.
gs_dump[user='omm'][localhost][port='20000'][testpdb][2024-04-26 16:57:19]: [100.00%] 459 objects have
been dumped.
gs_dump[user='omm'][localhost][port='20000'][testpdb][2024-04-26 16:57:19]: dump database testpdb
successfully
gs_dump[user='omm'][localhost][port='20000'][testpdb][2024-04-26 16:57:19]: total time: 5622  ms
```

**Example 10**

In the multi-tenancy scenario, use gs_dump to export the full information about the PDB named **testpdb**. The exported **backup_pdb_dir** file is in directory format.

```
gs_dump -U omm testpdb -p 20000 -f backup/backup_pdb_dir -F d
gs_dump[user='omm'][localhost][port='20000'][testpdb][2024-04-26 17:05:46]: The total objects number is
459.
gs_dump[user='omm'][localhost][port='20000'][testpdb][2024-04-26 17:05:46]: [100.00%] 459 objects have
been dumped.
gs_dump[user='omm'][localhost][port='20000'][testpdb][2024-04-26 17:05:46]: dump database testpdb
successfully
gs_dump[user='omm'][localhost][port='20000'][testpdb][2024-04-26 17:05:46]: total time: 5680  ms
```

**Example 11**

In the multi-tenancy scenario, when gs_dump is executed with the **-C, --create** option to export full information about the PDB named **testpdb**, gs_dump reports an error and exits.

```
gs_dump -U omm testpdb -C -p 20000 -f backup/backup_pdb_dir -F d
gs_dump unsupport the '-C, --create' option for pdb.
```

## Related Tools

**gs_dumpall for Exporting All Database Information** and **gs_restore for Importing Data**

## 2.4 gs_dumpall for Exporting All Database Information

### Description

gs_dumpall, provided by GaussDB, is used to export all database information, including the data of the default **postgres** database, data of user-specified databases, and global objects of all databases.

When gs_dumpall is used to export data, other users can still access (read or write) the database.

gs_dumpall can export complete, consistent data. For example, if gs_dumpall is started to export entire database at T1, data of the databases at that time point will be exported, and modifications on the databases after that time point will not be exported.

The generated columns are not dumped during gs_dumpall is used.

When gs_dump is used, the IMCV metadata (gs_imcv system catalog) created in the HTAP table is not dumped.

gs_dumpall exports all databases in two parts:

- gs_dumpall exports all global objects, including information about database users and groups, tablespaces, and attributes (for example, global access permissions).
- gs_dumpall calls gs_dump to export SQL scripts from each database, which contain all the SQL statements required to restore databases.

The exported files are both plain-text SQL scripts. Use **gsql** to execute them to restore databases.

gs_dumpall supports SSL encrypted communication. The method is the same as that of using gsql.

Before using gs_dumpall, ensure that the gs_dumpall version is consistent with the gs_dump version and database version. gs_dump and gs_dump of a later version may not be fully compatible with the kernel data of an earlier version.

**NOTE**

In M-compatible mode, do not import or export data between instances with different **lower_case_table_names** parameter settings. Otherwise, data may be lost.

## Precautions

- Do not modify any exported file. Otherwise, restoration may fail.
- To ensure the data consistency and integrity, gs_dumpall sets a share lock for a table to be dumped. If a share lock has been set for the table in other transactions, gs_dumpall locks the table after it is released. If the table cannot be locked within the specified time, the dump fails. You can customize the timeout interval to wait for lock release by specifying the **--lock-wait-timeout** parameter.
- During an export, gs_dumpall reads all tables in a database. Therefore, you need to connect to the database as a database administrator to export a complete file. When you use gsql to execute SQL scripts, administrator permissions are also required to add users and user groups, and create databases. Before importing a backup, you need to verify the security to prevent administrator permissions from being exploited.
- If you use gs_dumpall to export all database objects and want to import them to a new instance environment, ensure that the names and permissions of the users used for the export and import are the same. Otherwise, an error message will be displayed, indicating that the names are inconsistent or the permissions are insufficient.
- When files exported using gs_dumpall are imported to a new instance, the connected database must be the default database. Otherwise, syntax incompatibility may occur, resulting in import errors.

- Only one M-compatible database can be created globally. If an M-compatible database is exported, ensure that no other M-compatible database exists in the target instance environment.
- For scheduled tasks, this tool can export only scheduled tasks created using CREATE EVENT or non-periodic scheduled tasks created using advanced packages from a B-compatible database.
- gs_dumpall does not export user-defined tokenweight dictionaries. You can manually create the corresponding tokenweight dictionary according to the dictionary information displayed in the error message "WARNING: dictionary xx cannot be automatically exported, please create it manually."
- If the exported database contains PDBs, gs_dumpall does not export the template PDB and the created PDBs.
- When gs_dump is used, the IMCV metadata (gs_imcv system catalog) created in the HTAP table is not dumped.
- If transcoding is required for the specified dump encoding and data in the table contains invalid characters, the error message "invalid byte sequence" will be displayed during export. You are advised to specify the **-s** parameter when using gs_dump to export only definitions and execute **COPY** with the encoding fault tolerance option separately to export and import data.
- The current design of gs_dumpall is incompatible with separation of duties. In separation of duties, gs_dumpall can be used only by the initial user. The file to be imported must also be executed by the initial user.

## Format

```
gs_dumpall [OPTION]...
```

## Parameters

Common parameters:

- -f, --filename=<FILE_NAME>

  Sends the output to the specified file. If this option is omitted, the standard output is used.
- -v, --verbose

  Specifies the verbose mode. This will cause gs_dumpall to output detailed object comments and start/stop times to the dump file, and progress messages to standard errors.
- -V, --version

  Prints the gs_dumpall version and exits.
- --lock-wait-timeout=TIMEOUT

  Do not keep waiting to obtain shared table locks at the beginning of the dump. Consider it as failed if you are unable to lock a table within the specified time. The timeout interval can be specified in any of the formats accepted by SET statement_timeout.
- -?, --help

  Displays help about the command line parameters for gs_dumpall and exits.

Dump parameters:

- -a, --data-only

  Dumps only the data, not the schema (data definition).
- -c, --clean

  Runs SQL statements to delete databases before rebuilding them. Statements for dumping roles and tablespaces are added.
- -g, --globals-only

  Dumps only global objects (roles and tablespaces) but no databases.
- -o, --oids

  Dumps OIDs as parts of the data in each table. Use this option if your application references the OID columns in some way (for example, in a FOREIGN KEY constraint). If the preceding situation does not occur, do not use this parameter.
- -O, --no-owner

  Does not output commands to set ownership of objects to match the original database. By default, gs_dumpall issues the ALTER OWNER or SET SESSION AUTHORIZATION statement to set ownership of created schema elements. These statements will fail when the script is running unless it is started by a system administrator (or the same user who owns all of the objects in the script). By specifying **-O**, you can compile a script that can be stored by any user. The script grants the user the permission to own all objects because the ALTER OWNER or SET SESSION AUTHORIZATION statement is not used and the execute user permission is always used during the import. Therefore, before importing the dump file, check whether there are risks in the dump file. For example, check whether the dump file contains a privilege escalation statement and whether the statement is known to the administrator.
- -r, --roles-only

  Dumps only roles but not databases or tablespaces.
- -s, --schema-only

  Dumps only the object definition (schema) but not data.
- -S, --sysadmin=NAME

  Reserved for function extension. The option is not recommended.
- -t, --tablespaces-only

  Dumps only tablespaces but not databases or roles.
- -x, --no-privileges

  Prevents the dumping of access permissions (**GRANT**/**REVOKE** commands).
- --column-inserts/--attribute-inserts

  Exports data by running the **INSERT** command with explicit column names {INSERT INTO table (column, ...) VALUES ...}. This will cause a slow restoration. However, since this option generates an independent command for each row, an error in reloading a row causes only the loss of the row rather than the entire table content.

  **NOTE**

  M-compatible databases do not support this option. This option is skipped when an M-compatible database is exported.
- --disable-triggers

Reserved for function extension. The option is not recommended.

- --inserts

  Dumps data when the INSERT statement (rather than COPY) is issued. This will cause a slow restoration. The restoration may fail if you rearrange the column order. The **--column-inserts** parameter is safer against column order changes, though even slower.

  **NOTE**

  M-compatible databases do not support this option. This option is skipped when an M-compatible database is exported.

- --no-security-labels

  Reserved for function extension. The option is not recommended.

- --no-tablespaces

  Do not output statements to create tablespaces or select tablespaces for objects. All the objects will be created during the restoration process, no matter which tablespace is selected when using this option.

- --no-unlogged-table-data

  Reserved for function extension. The option is not recommended.

- --include-alter-table

  Exports information about deleted columns in the table.

- --quote-all-identifiers

  Forcibly quotes all identifiers. This parameter is useful when you dump a database for migration to a later version, in which additional keywords may be introduced.

  **NOTE**

  M-compatible databases do not support this option. This option is skipped when an M-compatible database is exported.

- --dont-overwrite-file

  Do not overwrite the current file.

- --use-set-session-authorization

  Specifies that the standard SQL **SET SESSION AUTHORIZATION** command rather than **ALTER OWNER** is generated to determine the object ownership. This makes dumping more standard. However, if a dump file contains objects that have historical problems, restoration may fail. A dump by running **SET SESSION AUTHORIZATION** requires the system administrator permissions, whereas running **ALTER OWNER** requires lower permissions. However, the SET SESSION AUTHORIZATION statement supports user and permission switching using a ciphertext password. The script exported using this parameter may not be restored. Therefore, you are advised not to use this parameter to export the script.

NOTE

Application scope of SET SESSION AUTHORIZATION:

- A system administrator can switch to a common user using the SET SESSION AUTHORIZATION statement, but cannot switch to the initial user, SYSADMIN, OPRADMIN, MONADMIN, POLADMIN, or AUDITADMIN.
- Other users cannot switch to another user using the SET SESSION AUTHORIZATION statement.

- --with-encryption=AES128

  Specifies that dumping data needs to be encrypted using AES-128.

- --with-key=KEY

  The AES-128 key rules are as follows:

  – Consists of 8 to 16 characters.

  – Contains at least three of the following character types: uppercase characters, lowercase characters, digits, and special characters (limited to ~!@#$ %^&*()-_=+\|[{}];:,<.>/?).

- --include-extensions

  Backs up all CREATE EXTENSION statements if the include-extensions parameter is set.

NOTICE

The extended function is for internal use only. You are advised not to use it.

- --include-templatedb

  Includes template databases during the dump.

- --binary-upgrade

  Reserved for function extension. The option is not recommended.

  NOTE

  M-compatible databases do not support this option. This option is skipped when an M-compatible database is exported.

- --binary-upgrade-usermap="USER1=USER2"
- Reserved for function extension. The option is not recommended.
- --non-lock-table

  This parameter is used only for inter-software API calling.

- --tablespaces-postfix

  Reserved for function extension. The option is not recommended.

- --parallel-jobs

  Specifies the number of concurrent backup processes. The value range is 1–1000.

- --pipeline

  Uses a pipe to transmit the password. This parameter cannot be used on devices.

NOTE

- The **-g/--globals-only** and **-r/--roles-only** parameters do not coexist.
- The **-g/--globals-only** and **-t/--tablespaces-only** parameters do not coexist.
- The **-r/--roles-only** and **-t/--tablespaces-only** parameters do not coexist.
- The **-s/--schema-only** and **-a/--data-only** parameters do not coexist.
- The **-r/--roles-only** and **-a/--data-only** parameters do not coexist.
- The **-t/--tablespaces-only** and **-a/--data-only** parameters do not coexist.
- The **-g/--globals-only** and **-a/--data-only** parameters do not coexist.
- **--tablespaces-postfix** must be used with **--binary-upgrade**.
- **--binary-upgrade-usermap** must be used with **--binary-upgrade**.
- **--parallel-jobs** must be used with **-f/--file**.

Connection parameters:

- -h, --host=HOSTNAME

  Specifies the host name. If the value begins with a slash (/), it is used as the UDS directory. The default value is taken from the *PGHOST* environment variable. If it is not set, a UDS connection is attempted.

  This parameter is used only for defining names of the hosts outside a database. For the localhost in the database, you can use IPv4 address **127.0.0.1** or IPv6 address **::1**.

  Environment variable: *PGHOST*
- -l, --database=DATABASENAME

  Specifies the name of the database connected to dump global objects and searches for other databases to be dumped. If this parameter is not specified, the **postgres** database will be dumped. If the **postgres** database does not exist, **template1** will be dumped.

  NOTE

  No matter which database is specified as the database to be connected for export, you must connect to **postgres** when using gsql to import data to a new instance. Otherwise, syntax incompatibility may occur, resulting in import errors. If the M-compatible database is incorrectly connected for import, the creation fails because CREATE DATABASE on the M-compatible database is equivalent to CREATE SCHEMA.
- -p, --port=PORT

  Specifies the TCP port listened on by the server or the local UDS file name extension to ensure a correct connection. The default value is the *PGPORT* environment variable.

  If the thread pool is enabled, you are advised to use the pooler port, that is, the listening port number plus 1.

  Environment variable: *PGPORT*
- -U, --username=NAME

  Specifies the name of the connected user. The initial user cannot be used for cross-node execution.

  Environment variable: *PGUSER*
- -w, --no-password

  Never issues a password prompt. The connection attempt fails if the server requires password for authentication and the password is not provided in other ways. This parameter is useful in batch jobs and scripts in which no user password is required.

- -W, --password=PASSWORD

  Specifies the user password for connection. If the authentication policy of the host is **trust**, the password of the system administrator is not verified. That is, you do not need to enter the **-W** option. If this parameter is not specified and you are not a system administrator, the system prompts you to enter the password in interactive mode. To ensure system security, you are advised to enter the password in interactive mode.
- --role=ROLENAME

  Specifies a role name to be used for creating the dump. This option causes gs_dumpall to issue the SET ROLE statement after connecting to the database. It is useful when the authenticated user (specified by **-U**) lacks the permissions required by gs_dumpall. It allows the user to switch to a role with the required permissions. Some installations have a policy against logging in directly as a system administrator. This option allows dumping data without violating the policy.
- --rolepassword=ROLEPASSWORD

  Specifies the password of a specific role or user.

## Notes

- gs_dumpall internally calls gs_dump. For details about the diagnosis information, see **gs_dump for Exporting Database Information**.
- After restoration, it is advised to run **ANALYZE** on each database so that the optimizer can provide useful statistics.
- gs_dumpall requires all needed tablespace directories to be empty before the restoration. Otherwise, database creation will fail if the databases are in non-default locations.

## Example

Use gs_dumpall to export all databases at a time.

NOTE

gs_dumpall supports only plain-text format export. Therefore, only gsql can be used to restore a file exported using gs_dumpall.

- Export all databases.

  **backup/bkp2.sql** indicates the exported file, **29900** indicates the port number of the database server, and **omm** indicates the username.

```
gs_dumpall -U omm -f ./backup/bkp2.sql -p 29900
gs_dump[user='omm'][localhost][port='29900'][dbname='postgres'][2024-11-11 11:00:01]: The total
objects number is 458.
gs_dump[user='omm'][localhost][port='29900'][dbname='postgres'][2024-11-11 11:00:01]: [100.00%]
458 objects have been dumped.
gs_dump[user='omm'][localhost][port='29900'][dbname='postgres'][2024-11-11 11:00:01]: dump
database dbname='postgres' successfully
gs_dump[user='omm'][localhost][port='29900'][dbname='postgres'][2024-11-11 11:00:01]: total time:
3239  ms
gs_dumpall[user='omm'][localhost][port='29900'][2024-11-11 11:00:01]: dumpall operation successful
gs_dumpall[user='omm'][localhost][port='29900'][2024-11-11 11:00:01]: total time: 3305  ms
```

- Specify four jobs to concurrently export all databases.

```
gs_dumpall -U omm -f ./backup/bkp2.sql -p 29900 --parallel-jobs 4
gs_dump[user='omm'][localhost][port='29900'][dbname='postgres'][2024-11-11 11:04:13]: The total objects number is 458.
gs_dump[user='omm'][localhost][port='29900'][dbname='postgres'][2024-11-11 11:04:13]: [100.00%] 458 objects have been dumped.
gs_dump[user='omm'][localhost][port='29900'][dbname='postgres'][2024-11-11 11:04:13]: dump database dbname='postgres' successfully
gs_dump[user='omm'][localhost][port='29900'][dbname='postgres'][2024-11-11 11:04:13]: total time: 3261  ms
gs_dumpall[user='omm'][localhost][port='29900'][2024-11-11 11:04:13]: dumpall operation successful
gs_dumpall[user='omm'][localhost][port='29900'][2024-11-11 11:04:13]: total time: 3320  ms
```

- Export only global objects (roles and tablespaces).

```
gs_dumpall -U omm -f ./backup/bkp2.sql -p 29900 -g
gs_dumpall[user='omm'][localhost][port='29900'][2024-11-11 11:06:00]: dumpall operation successful
gs_dumpall[user='omm'][localhost][port='29900'][2024-11-11 11:06:00]: total time: 45  ms
```

- Encrypt the exported files using AES-128.

  Replace ******** with the actual password.

```
gs_dumpall -U omm -f ./backup/bkp2.sql -p 29900 --with-encryption=AES128 --with-key='********'
gs_dump[user='omm'][localhost][port='29900'][dbname='postgres'][2024-11-11 11:07:20]: The total objects number is 459.
gs_dump[user='omm'][localhost][port='29900'][dbname='postgres'][2024-11-11 11:07:20]: [100.00%] 459 objects have been dumped.
gs_dump[user='omm'][localhost][port='29900'][dbname='postgres'][2024-11-11 11:07:21]: dump database dbname='postgres' successfully
gs_dump[user='omm'][localhost][port='29900'][dbname='postgres'][2024-11-11 11:07:21]: total time: 3518  ms
gs_dumpall[user='omm'][localhost][port='29900'][2024-11-11 11:07:21]: dumpall operation successful
gs_dumpall[user='omm'][localhost][port='29900'][2024-11-11 11:07:21]: total time: 3764  ms
```

### Related Commands

**gs_dump for Exporting Database Information**

## 2.5 gs_restore for Importing Data

### Description

gs_restore, provided by GaussDB, is used to import data that was exported using gs_dump. It can also be used to import files exported by gs_dump.

It has the following functions:

- Imports data to the database.

  If a database is specified, data is imported to the database. For parallel import, the password for connecting to the database is required. During data import, the generated columns are automatically updated and saved as ordinary columns.

  If a PDB is specified in the connection parameters, data is imported to the specified PDB.

NOTE

- If data is imported to a specified PDB, the **-C, --create** option is not supported.
- If the specified PDB is in the **close** state, the import fails.
- If the transaction of importing data to a specified PDB is set to read-only, the import fails.
- In M-compatible mode, do not import or export data between instances with different **lower_case_table_names** parameter settings. Otherwise, data may be lost.
- The current design of gs_restore is incompatible with separation of duties. In separation of duties, gs_restore can be used only by the initial user. The file to be imported must also be executed by the initial user.

- Imports data to an archive file.

  If the **-l** parameter is specified, an archive file containing a brief summary of the data is generated.

gs_restore supports SSL encrypted communication. The method is the same as that of using gsql.

## Prerequisites

Before using gs_restore, ensure that the gs_restore version is consistent with the gs_dump version and database version.

## Format

```
gs_restore [OPTION]... FILE
```

NOTE

- The **FILE** does not have a short or long option. It is used to specify the location for the archive files.
- The **dbname** or **-l** option is required as prerequisites. Users cannot enter **dbname** and **-l** parameters at the same time.
- gs_restore incrementally imports data by default. To prevent data exception caused by consecutive imports, use the **-e** and **-c** parameters for each import. **-c** indicates that the database objects that already exist in the database to be restored are cleared (deleted) before the database objects are rebuilt. **-e** indicates that if an error occurs when an SQL statement is sent to the database, the system exits. By default, the system continues to import data and displays a series of error information after the import is complete.
- If the owner of the schema object has the OPRADMIN system permission, the initial user is required for the import.
- During data import, if invalid characters are found and do not need to be transcoded, you can set the database compatibility parameter **copy_special_character_version** to **'no_error'** to import the data. Otherwise, an error is reported but the import is not interrupted.

## Parameters

Common parameters:

- -d, --dbname=NAME

  Connects to the **dbname** database and imports data to the database.

- -f, --file=<FILE_NAME>

Specifies the output file for the generated archive, or the output file in the list specified using **-l**.

The default is the standard output.

NOTE

**-f** cannot be used with **-d**.

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

  Specifies the format of the archive. The format does not need to be specified because the gs_restore determines the format automatically.

  Value range:

  – **c/custom**: The archive form is the customized format in gs_dump.

  – **d/directory**: The archive format is **directory**.

  – **t/tar**: The archive format is **tar**.

- -l, --list

  Lists the formats of the archive. The operation output can be used for the input of the **-L** option. If filtering parameters, such as **-n** or **-t**, are used together with **-l**, they will restrict the listed items.

- -v, --verbose

  Specifies the verbose mode.

- -V, --version

  Prints the gs_restore version and exits.

- -?, --help

  Displays help information about gs_restore parameters and exits.

Import parameters:

- -a, -data-only

  Imports only the data, not the schema (data definition). gs_restore incrementally imports data.

- -c, --clean

  Cleans (deletes) existing database objects in the database to be restored before recreating them. If the target database does not contain the objects involved in the deletion operation, some promptive error information may be displayed.

- -C, --create

  Before data is imported to a database, CREATE DATABASE is used to create the database. (After this option is specified, the database specified by **-d** is only used to execute the **CREATE DATABASE** command, and all data is still imported to the created database.)

  NOTE

  - In multi-tenancy scenarios, this option is not supported when gs_restore is used to import data to a PDB.
  - This option is not supported in M-compatible databases. You must create a database in M-compatible mode on the target instance, export data from the source instance, and connect to the newly created M-compatible database on the target instance to import data.

- -e, --exit-on-error

  Exits if an error occurs when you send the SQL statement to the database. If you do not exit, the commands will still be sent and error information will be displayed when the import ends.

- -I, --index=NAME

  Imports only the definition of the specified index. Multiple indexes can be imported. Enter *-I index* multiple times to import multiple indexes.

  For example:

  ```
  gs_restore -h host_name -p port_number -d testdb -I Index1 -I Index2 backup/MPPDB_backup.tar
  ```

  In this example, **Index1** and **Index2** will be imported.

- -j, --jobs=NUM

  Specifies the number of concurrent, the most time-consuming jobs of gs_restore (such as loading data, creating indexes, or creating constraints). This parameter can greatly reduce the time to import a large database to a server running on a multi-processor machine.

  Each task may be a process or a thread, which is determined by the OS. Each task is connected to the server separately.

  The optimal value for this option depends on the server hardware settings, the client, the network, the number of CPU cores, and disk settings. It is recommended that the parameter be set to the number of CPU cores on the server. In addition, a larger value can also lead to faster import in many cases. However, an overly large value will lead to decreased performance because of thrashing.

  This option supports the customized archive format only. The input file must be a regular file (not the pipe file). This parameter can be ignored when you select the script method rather than connecting to a database server. In addition, multiple jobs cannot be used with the **--single-transaction** option.

  **NOTE**

  - This parameter applies to multi-table, multi-index, and multi-constraint scenarios. In practice, the number of created processes (or threads) is related to the number of tables, indexes, and constraints. The maximum number of concurrent jobs does not exceed the specified number of jobs.
  - When gs_restore imports data to a PDB, the optimal value of this parameter depends on the resources allocated to the PDB.

- -L, --use-list=<FILE_NAME>

  Imports only archive elements that are listed in **list-file** and imports them in the order that they appear in the file. If filtering parameters, such as **-n** or **-t**, are used with **-L**, they will further limit the items to be imported.

  **list-file** is normally created by editing the output of a previous **-l** parameter. File lines can be modified or removed, and can also be commented out by placing a semicolon (;) at the beginning of the row.

- -n, --schema=NAME

  Restores only objects that are listed in schemas.

  This parameter can be used with the **-t** parameter to import a specific table.

  Entering **-n** *schemaname* multiple times can import multiple schemas.

  For example:

  ```
  gs_restore -h host_name -p port_number -d testdb -n sch1 -n sch2 backup/MPPDB_backup.tar
  ```

In this example, **sch1** and **sch2** will be imported.

NOTE

- In M-compatible mode, if a database with **templatem** is created by running **CREATE DATABASE**, data is imported by specifying **db_name** using the **-d** option; if a database is created by running **CREATE DATABASE** *db_name*, data is imported by specifying **-n** because such database is equivalent to a schema.
- The specified schema name must exist in the archive file that is used as the input. If a schema name that does not exist in the archive file is specified, the import of the schema does not take effect.
- Different from the case in gs_dump, this parameter is fully matched in gs_restore. When the schema name contains uppercase letters, no extra quotation marks are required, for example, **-n "Sch1"**.
- In M-compatible mode, the value of this parameter is affected by the GUC parameter **lower_case_table_names**. For example, in case-sensitive mode (**lower_case_table_names** set to **0**), the value of this parameter must be case-sensitive. If the value contains uppercase letters, no extra quotation marks are required; in case-insensitive mode (**lower_case_table_names** set to **1**), the value of this parameter must be in lowercase.

- -O, --no-owner

  Do not output commands to set ownership of objects to match the original database. By default, gs_restore issues the ALTER OWNER or SET SESSION AUTHORIZATION statement to set ownership of created schema elements. Unless the system administrator or the user who has all the objects in the script initially connects to the database. Otherwise, the statement will fail. Any username can be used for the initial connection using **-O**, and this user will own all the created objects.

- -P, --function=NAME(args)

  Imports only listed functions. You need to correctly spell the function name and the parameter based on the contents of the dump file in which the function exists.

  Entering **-P** alone means importing all function-name(args) functions in a file. Entering **-P** with **-n** means importing the function-name(args) functions in a specified schema. Entering **-P** multiple times and using **-n** once means that all imported functions are in the **-n** schema by default.

  You can enter **-n schema-name -P 'function-name(args)'** multiple times to import functions in specified schemas.

  For example:

  ```
  ./gs_restore -h host_name -p port_number -d testdb -n test1 -P 'Func1(integer)' -n test2 -P
  'Func2(integer)' backup/MPPDB_backup.tar
  ```

  In this example, both **Func1 (i integer)** in the **test1** schema and **Func2 (j integer)** in the **test2** schema will be imported.

- -s, --schema-only

  Imports only schemas (data definitions), instead of data (table content). The current sequence value will not be imported.

- -S, --sysadmin=NAME

  Reserved for function extension. The option is not recommended.

- -t, --table=NAME

  Imports only listed table definitions or data, or both. This parameter can be used with the **-n** parameter to specify a table object in a schema. When **-n** is not entered, the default schema is **PUBLIC**. Entering **-n** *schemaname* **-t** *tablename* multiple times can import multiple tables in a specified schema.

For example:

Import **table1** in the **PUBLIC** schema.

```
gs_restore -h host_name -p port_number -d testdb -t table1 backup/MPPDB_backup.tar
```

Import **test1** in the **test1** schema and **test2** in the **test2** schema.

```
gs_restore -h host_name -p port_number -d testdb -n test1 -t test1 -n test2 -t test2 backup/
MPPDB_backup.tar
```

Import **table1** in the **PUBLIC** schema and **test1** in the **test1** schema.

```
gs_restore -h host_name -p port_number -d testdb -n PUBLIC -t table1 -n test1 -t table1 backup/
MPPDB_backup.tar
```

NOTICE

- **-t** does not support the *schema_name.table_name* input format. If this format is specified, no error is reported but the setting does not take effect.
- When **-t** is specified, gs_restore does not import any other database objects that are attached to the selected table. Therefore, there is no guarantee that the results of a specific-table dump can be automatically imported to an empty database.
- **-t tablename** does not import trigger information from a table.
- Different from the case in gs_dump, this parameter is fully matched in gs_restore. When the table name contains uppercase letters, no extra quotation marks are required, for example, **-t "Tbl1"**.
- In M-compatible mode, the value of this parameter is affected by the GUC parameter **lower_case_table_names**. For example, in case-sensitive mode (**lower_case_table_names** set to **0**), the value of this parameter must be case-sensitive. If the value contains uppercase letters, no extra quotation marks are required; in case-insensitive mode (**lower_case_table_names** set to **1**), the value of this parameter must be in lowercase.

- -T, --trigger=NAME

  This parameter is reserved for extension.
- -x, --no-privileges/--no-acl

  Prevents the import of access permissions (**GRANT**/**REVOKE** commands).
- -1, --single-transaction

  Executes import as a single transaction (that is, commands are wrapped in **BEGIN**/**COMMIT**).

  This option ensures that either all the commands are completed successfully, or no application is changed. This option means **--exit-on-error**.
- --disable-triggers

  Reserved for function extension. The option is not recommended.
- --no-data-for-failed-tables

  By default, table data will be imported even if the statement to create a table fails (for example, the table exists). Data in such table is skipped using this parameter. This operation is useful if the target database already contains the desired table contents.

This parameter takes effect only when you import data directly to a database, not when you output SQL scripts.

- --no-security-labels

  Reserved for function extension. The option is not recommended.

- --no-tablespaces

  Does not select any tablespaces. All the objects will be created during the import process, no matter which tablespace is selected when using this option.

- --section=SECTION

  Imports the listed sections (such as pre-data, data, or post-data).

- --use-set-session-authorization

  This option is used for backing up the plain-text format.

  Generates the **SET SESSION AUTHORIZATION** statement as the output instead of the **ALTER OWNER** statement to determine object ownership. This parameter makes dump more standards-compatible. If the records of objects in exported files are referenced, import may fail. Only administrators can use the **SET SESSION AUTHORIZATION** statement to dump data, and the administrators must manually change and verify the passwords of exported files by referencing the **SET SESSION AUTHORIZATION** statement before import. The **ALTER OWNER** statement requires lower permissions.

  NOTE

  If the **--use-set-session-authorization** parameter is used by a system administrator to import data and the imported data contains objects of the initial user, "ERROR: permission denied to set session authorization" is reported and the import is interrupted when the system administrator switches to the initial user due to security permission verification. In this scenario, you are advised to use the initial user to import and export data.

- --pipeline

  Uses a pipe to transmit the password. This parameter cannot be used on devices.

NOTICE

- If any local additions need to add to the template1 database during the installation, restore the output of gs_restore into an empty database with caution. Otherwise, you are likely to obtain errors due to duplicate definitions of the added objects. To create an empty database without any local additions, copy data from template0 rather than template1. For example:

```
gaussdb=# CREATE DATABASE foo WITH TEMPLATE template0;
```

- gs_restore cannot import large objects selectively. For example, it can only import the objects of a specified table. If an archive form contains large objects, all large objects will be imported. If this archive object is excluded by **-L**, **-t**, or other options, none of the large objects will be imported.

NOTE

- The **-d/--dbname** and **-f/--file** options do not coexist.
- The **-s/--schema-only** and **-a/--data-only** options do not coexist.
- The **-c/--clean** and **-a/--data-only** options do not coexist.
- When the **--single-transaction** option is used, **-j/--jobs** must be a single job.
- **--role** must be used with **--rolepassword**.

Connection parameters:

- -h, --host=HOSTNAME

  Specifies the host name. If the value begins with a slash (/), it is used as the UDS directory. The default value is taken from the *PGHOST* environment variable. If it is not set, a UDS connection is attempted.

  This parameter is used only for defining names of the hosts outside a database. For the localhost in the database, you can use IPv4 address **127.0.0.1** or IPv6 address **::1**.

  Environment variable: *PGHOST*
- -p, --port=PORT

  Specifies the TCP port listened on by the server or the local UDS file name extension to ensure a correct connection. The default value is the *PGPORT* environment variable.

  If the thread pool is enabled, you are advised to use pooler port, that is, the listening port number plus 1.

  Environment variable: *PGPORT*
- -U, --username=NAME

  Specifies the name of the connected user. The initial user cannot be used for cross-node execution.

  Environment variable: *PGUSER*
- -w, --no-password

  Never issues a password prompt. The connection attempt fails if the server requires password for authentication and the password is not provided in other ways. This parameter is useful in batch jobs and scripts in which no user password is required.
- -W, --password=PASSWORD

  Specifies the user password for connection. If the authentication policy of the host is **trust**, the password of the system administrator is not verified. That is, you do not need to enter the **-W** parameter. If this parameter is not specified and you are not a system administrator, the system prompts you to enter the password in interactive mode. To ensure system security, you are advised to enter the password in interactive mode.
- --role=ROLENAME

  Specifies a role name for the import operation. If this parameter is selected, the SET ROLE statement will be issued after gs_restore connects to the database. It is useful when the authenticated user (specified by **-U**) lacks the permissions required by gs_restore. This parameter allows the user to switch to a role with the required permissions. Some installations have a policy against logging in directly as the initial user. This parameter allows data to be imported without violating the policy.

- --rolepassword=ROLEPASSWORD

  Specifies the password of the specific role.

## Examples

gs_restore is used to import the files exported by gs_dump.

- **Example 1**:

  Execute the gs_restore tool to import the exported **MPPDB_backup.dmp** file (custom format) to the **testdb** database.

  ```
  gs_restore backup/MPPDB_backup.dmp -p 8000 -d testdb
  restore operation successful
  total time: 13053  ms
  ```

- **Example 2**:

  Execute the gs_restore tool to import the exported **MPPDB_backup.tar** file (tar format) to the **testdb** database.

  ```
  gs_restore backup/MPPDB_backup.tar -p 8000 -d testdb
  restore operation successful
  total time: 21203  ms
  ```

- **Example 3**:

  Execute the gs_restore tool to import the exported **MPPDB_backup** file (directory format) to the **testdb** database.

  ```
  gs_restore backup/MPPDB_backup -p 8000 -d testdb
  restore operation successful
  total time: 21003  ms
  ```

- **Example 4**:

  Execute the gs_restore tool and import the **MPPDB_backup.dmp** file (custom format). Specifically, import all the object definitions and data in the **PUBLIC** schema. Existing objects are deleted from the target database before the import. If an existing object references to an object in another schema, you need to manually delete the referenced object first.

  ```
  gs_restore backup/MPPDB_backup.dmp -p 8000 -d testdb -e -c -n PUBLIC
  Error while PROCESSING TOC:
  Error from TOC entry 313; 1259 337399 TABLE table1 gaussdba
  could not execute query: ERROR:  cannot drop table table1 because other objects depend on it
  DETAIL:  view t1.v1 depends on table table1
  HINT:  Use DROP ... CASCADE to drop the dependent objects too.
      Command was: DROP TABLE IF EXISTS public.table1;
  ```

  Manually delete the referenced object and create it again after the import is complete.

  ```
  gs_restore backup/MPPDB_backup.dmp -p 8000 -d testdb -e -c -n PUBLIC
  restore operation successful
  total time: 2203  ms
  ```

- **Example 5**:

  Execute the gs_restore tool and import the **MPPDB_backup.dmp** file (custom format). Specifically, import only the definition of **table1** in the **PUBLIC** schema.

  ```
  gs_restore backup/MPPDB_backup.dmp -p 8000 -d testdb -e -c -s -n PUBLIC -t table1
  restore operation successful
  total time: 21000  ms
  ```

- **Example 6**:

  Execute the gs_restore tool and import the **MPPDB_backup.dmp** file (custom format). Specifically, import only the data of **table1** in the **PUBLIC** schema.

```
gs_restore backup/MPPDB_backup.dmp -p 8000 -d testdb -e -a -n PUBLIC -t table1
restore operation successful
total time: 20203  ms
```

- **Example 7**:

  In the multi-tenancy scenario, use gs_restore to import the specified **PDB_backup.dmp** file (user-defined archive format) to the PDB named **testpdb**.

  ```
  gs_restore backup/PDB_backup.dmp -p 20000 -d testpdb
  restore operation successful
  total time: 371  ms
  ```

- **Example 8**:

  In the multi-tenancy scenario, use gs_restore to import the specified **PDB_backup.tar** file (tar format) to the PDB named **testpdb**.

  ```
  gs_restore backup/PDB_backup.tar -p 20000 -d testpdb
  restore operation successful
  total time: 367  ms
  ```

- **Example 9:**

  In the multi-tenancy scenario, use gs_restore to import the specified **PDB_backup** directory file (directory archive format) to the PDB named **testpdb**.

  ```
  gs_restore backup/PDB_backup -p 20000 -d testpdb
  restore operation successful
  total time: 370  ms
  ```

- **Example 10:**

  In the multi-tenancy scenario, use gs_restore with the **-C, --create** option to import data. gs_restore reports an error and exits.

  ```
  gs_restore backup/PDB_backup -C -p 20000 -d testpdb
  gs_restore unsupport the '-C, --create' option for pdb.
  ```

- **Special case**:

  Execute the gsql tool. Import the **MPPDB_backup.sql** file in the export folder (plain-text format) generated by gs_dump or gs_dumpall to the **testdb** database.

  ```
  gsql -d testdb -p 8000 -f /home/omm/test/MPPDB_backup.sql
  SET
  SET
  SET
  SET
  SET
  ALTER TABLE
  ALTER TABLE
  ALTER TABLE
  ALTER TABLE
  ALTER TABLE
  CREATE INDEX
  CREATE INDEX
  CREATE INDEX
  SET
  CREATE INDEX
  REVOKE
  REVOKE
  GRANT
  GRANT
  total time: 30476  ms
  ```

  In the example, the file after **-f** is the exported file, and **8000** indicates the port number of the database server. **testdb** indicates the name of the database to be accessed.

## Related Commands

**gs_dump for Exporting Database Information** and **gs_dumpall for Exporting All Database Information**