Rocky Enterprise Linux 9.2 Manual Pages on command 'mysql.1'

$ man mysql.1

MYSQL(1) MySQL Database System MYSQL(1)

NAME

mysql - the MySQL command-line client

SYNOPSIS

mysql [options] db_name

DESCRIPTION

mysql is a simple SQL shell with input line editing capabilities. It supports interactive and noninteractive use. When used interactively, query results are presented in an ASCII-table format. When used noninteractively (for example, as a filter), the result is presented in tab-separated format. The output format can be changed using command options.

If you have problems due to insufficient memory for large result sets, use the --quick option. This forces mysql to retrieve results from the server a row at a time rather than retrieving the entire result set and buffering it in memory before displaying it. This is done by returning the result set using the mysql_use_result() C API function in the client/server library rather than mysql_store_result().

Note

Alternatively, MySQL Shell offers access to the X DevAPI. For details, see MySQL Shell 8.0[1].

Using mysql is very easy. Invoke it from the prompt of your command interpreter as follows:

mysql db_name

Or:

mysql --user=user_name --password db_name

In this case, you'll need to enter your password in response to the prompt that mysql displays:

Enter password: your_password

Then type an SQL statement, end it with ;, \g, or \G and press Enter.

Typing Control+C interrupts the current statement if there is one, or cancels any partial input line otherwise.

You can execute SQL statements in a script file (batch file) like this:

mysql db_name < script.sql > output.tab

On Unix, the mysql client logs statements executed interactively to a history file. See the section called ?MYSQL CLIENT LOGGING?.

MYSQL CLIENT OPTIONS

mysql supports the following options, which can be specified on the command line or in the [mysql] and [client] groups of an option file. For information about option files used by MySQL programs, see Section 4.2.2.2, ?Using Option Files?.

? --help, -? Display a help message and exit.

? --auto-rehash Enable automatic rehashing. This option is on by default, which enables database, table, and column name completion. Use --disable-auto-rehash to disable rehashing. That causes mysql to start faster, but you must issue the rehash command or its \# shortcut if you want to use name completion.

To complete a name, enter the first part and press Tab. If the name is unambiguous, mysql completes it. Otherwise, you can press Tab again to see the possible names that begin with what you have typed so far. Completion does not occur if there is no default database.

Note

This feature requires a MySQL client that is compiled with the readline library. Typically, the readline library is not available on Windows.

? --auto-vertical-output Cause result sets to be displayed vertically if they are too wide for the current window, and using normal tabular format otherwise. (This applies to statements terminated by ; or \G.)

? --batch, -B Print results using tab as the column separator, with each row on a new line. With this option, mysql does not use the history file.

Batch mode results in nontabular output format and escaping of special characters. Escaping may be disabled by using raw mode; see the description for the --raw option.

? --binary-as-hex When this option is given, mysql displays binary data using hexadecimal notation (0xvalue). This occurs whether the overall output display format is tabular, vertical, HTML, or XML.

--binary-as-hex when enabled affects display of all binary strings, including those returned by functions such as CHAR() and UNHEX().

The following example demonstrates this using the ASCII code for A (65 decimal, 41 hexadecimal):

? --binary-as-hex disabled:

```
mysql> SELECT CHAR(0x41), UNHEX('41');
+------------+-------------+
| CHAR(0x41) | UNHEX('41') |
+------------+-------------+
| A          | A           |
+------------+-------------+
```

? --binary-as-hex enabled:

```
mysql> SELECT CHAR(0x41), UNHEX('41');
+------------------------+--------------------------+
| CHAR(0x41)             | UNHEX('41')              |
+------------------------+--------------------------+
```

| 0x41 | 0x41 |

+----------+----------+

To write a binary string expression so that it displays as a character string regardless of whether --binary-as-hex is enabled, use these techniques:

? The CHAR() function has a USING charset clause:

```
mysql> SELECT CHAR(0x41 USING utf8mb4);
+--------------------------+
| CHAR(0x41 USING utf8mb4) |
+--------------------------+
| A                        |
+--------------------------+
```

? More generally, use CONVERT() to convert an expression to a given character set:

```
mysql> SELECT CONVERT(UNHEX('41') USING utf8mb4);
+-----------------------------------+
| CONVERT(UNHEX('41') USING utf8mb4) |
+-----------------------------------+
| A                                 |
+-----------------------------------+
```

As of MySQL 8.0.19, when mysql operates in interactive mode, this option is enabled by default. In addition, output from the status (or \s) command includes this line when the option is enabled implicitly or explicitly:

Binary data as: Hexadecimal

To disable hexadecimal notation, use --skip-binary-as-hex

? --binary-mode This option helps when processing mysqlbinlog output that may contain BLOB values. By default, mysql translates \r\n in statement strings to \n and interprets \0 as the statement terminator. --binary-mode disables both features. It also disables all mysql commands except charset and delimiter in noninteractive mode (for input piped to mysql or loaded using the source command).

? --bind-address=ip_address On a computer having multiple network

interfaces, use this option to select which interface to use for connecting to the MySQL server.

? --character-sets-dir=dir_name The directory where character sets are installed. See Section 10.15, ?Character Set Configuration?.

? --column-names Write column names in results.

? --column-type-info Display result set metadata. This information corresponds to the contents of C API MYSQL_FIELD data structures. See C API Basic Data Structures[2].

? --comments, -c Whether to strip or preserve comments in statements sent to the server. The default is --skip-comments (strip comments), enable with --comments (preserve comments).

Note

The mysql client always passes optimizer hints to the server, regardless of whether this option is given.

Comment stripping is deprecated. Expect this feature and the options to control it to be removed in a future MySQL release.

? --compress, -C Compress all information sent between the client and the server if possible. See Section 4.2.8, ?Connection Compression Control?.

As of MySQL 8.0.18, this option is deprecated. Expect it to be removed in a future version of MySQL. See the section called ?Configuring Legacy Connection Compression?.

? --compression-algorithms=value The permitted compression algorithms for connections to the server. The available algorithms are the same as for the protocol_compression_algorithms system variable. The default value is uncompressed.

For more information, see Section 4.2.8, ?Connection Compression Control?.

This option was added in MySQL 8.0.18.

? --connect-expired-password Indicate to the server that the client can handle sandbox mode if the account used to connect has an expired password. This can be useful for noninteractive invocations of mysql because normally the server disconnects noninteractive

clients that attempt to connect using an account with an expired password. (See Section 6.2.16, ?Server Handling of Expired Passwords?.)

? --connect-timeout=value The number of seconds before connection timeout. (Default value is 0.)

? --database=db_name, -D db_name The database to use. This is useful primarily in an option file.

? --debug[=debug_options], -# [debug_options] Write a debugging log. A typical debug_options string is d:t:o,file_name. The default is d:t:o,/tmp/mysql.trace.
This option is available only if MySQL was built using WITH_DEBUG. MySQL release binaries provided by Oracle are not built using this option.

? --debug-check Print some debugging information when the program exits.
This option is available only if MySQL was built using WITH_DEBUG. MySQL release binaries provided by Oracle are not built using this option.

? --debug-info, -T Print debugging information and memory and CPU usage statistics when the program exits.
This option is available only if MySQL was built using WITH_DEBUG. MySQL release binaries provided by Oracle are not built using this option.

? --default-auth=plugin A hint about which client-side authentication plugin to use. See Section 6.2.17, ?Pluggable Authentication?.

? --default-character-set=charset_name Use charset_name as the default character set for the client and connection.
This option can be useful if the operating system uses one character set and the mysql client by default uses another. In this case, output may be formatted incorrectly. You can usually fix such issues by using this option to force the client to use the system character set instead.
For more information, see Section 10.4, ?Connection Character Sets

and Collations?, and Section 10.15, ?Character Set Configuration?.

? --defaults-extra-file=file_name Read this option file after the global option file but (on Unix) before the user option file. If the file does not exist or is otherwise inaccessible, an error occurs. If file_name is not an absolute path name, it is interpreted relative to the current directory.
For additional information about this and other option-file options, see Section 4.2.2.3, ?Command-Line Options that Affect Option-File Handling?.

? --defaults-file=file_name Use only the given option file. If the file does not exist or is otherwise inaccessible, an error occurs. If file_name is not an absolute path name, it is interpreted relative to the current directory.
Exception: Even with --defaults-file, client programs read .mylogin.cnf.
For additional information about this and other option-file options, see Section 4.2.2.3, ?Command-Line Options that Affect Option-File Handling?.

? --defaults-group-suffix=str Read not only the usual option groups, but also groups with the usual names and a suffix of str. For example, mysql normally reads the [client] and [mysql] groups. If this option is given as --defaults-group-suffix=_other, mysql also reads the [client_other] and [mysql_other] groups.
For additional information about this and other option-file options, see Section 4.2.2.3, ?Command-Line Options that Affect Option-File Handling?.

? --delimiter=str Set the statement delimiter. The default is the semicolon character (;).

? --disable-named-commands Disable named commands. Use the * form only, or use named commands only at the beginning of a line ending with a semicolon (;). mysql starts with this option enabled by default. However, even with this option, long-format commands still work from the first line. See the section called ?MYSQL CLIENT

COMMANDS?.

? --dns-srv-name=name Specifies the name of a DNS SRV record that determines the candidate hosts to use for establishing a connection to a MySQL server. For information about DNS SRV support in MySQL, see Section 4.2.6, ?Connecting to the Server Using DNS SRV Records?.

Suppose that DNS is configured with this SRV information for the example.com domain:

```
Name                    TTL   Class  Priority Weight Port Target
_mysql._tcp.example.com. 86400 IN SRV  0        5      3306 host1.example.com
_mysql._tcp.example.com. 86400 IN SRV  0        10     3306 host2.example.com
_mysql._tcp.example.com. 86400 IN SRV  10       5      3306 host3.example.com
_mysql._tcp.example.com. 86400 IN SRV  20       5      3306 host4.example.com
```

To use that DNS SRV record, invoke mysql like this:

```
mysql --dns-srv-name=_mysql._tcp.example.com
```

mysql then attempts a connection to each server in the group until a successful connection is established. A failure to connect occurs only if a connection cannot be established to any of the servers.

The priority and weight values in the DNS SRV record determine the order in which servers should be tried.

When invoked with --dns-srv-name, mysql attempts to establish TCP connections only.

The --dns-srv-name option takes precedence over the --host option if both are given. --dns-srv-name causes connection establishment to use the mysql_real_connect_dns_srv() C API function rather than mysql_real_connect(). However, if the connect command is subsequently used at runtime and specifies a host name argument, that host name takes precedence over any --dns-srv-name option given at mysql startup to specify a DNS SRV record.

This option was added in MySQL 8.0.22.

? --enable-cleartext-plugin Enable the mysql_clear_password cleartext authentication plugin. (See Section 6.4.1.4, ?Client-Side Cleartext Pluggable Authentication?.)

? --execute=statement, -e statement Execute the statement and quit.

The default output format is like that produced with --batch. See Section 4.2.2.1, ?Using Options on the Command Line?, for some examples. With this option, mysql does not use the history file.

? --fido-register-factor=value The factor or factors for which FIDO device registration must be performed. This option value must be a single value, or two values separated by commas. Each value must be 2 or 3, so the permitted option values are '2', '3', '2,3' and '3,2'.

For example, an account that requires registration for a 3rd authentication factor invokes the mysql client as follows:

```
mysql --user=user_name --fido-register-factor=3
```

An account that requires registration for a 2nd and 3rd authentication factor invokes the mysql client as follows:

```
mysql --user=user_name --fido-register-factor=2,3
```

If registration is successful, a connection is established. If there is an authentication factor with a pending registration, a connection is placed into pending registration mode when attempting to connect to the server. In this case, disconnect and reconnect with the correct --fido-register-factor value to complete the registration.

Registration is a two step process comprising initiate registration and finish registration steps. The initiate registration step executes this statement:

```
ALTER USER user factor INITIATE REGISTRATION
```

The statement returns a result set containing a 32 byte challenge, the user name, and the relying party ID (see authentication_fido_rp_id).

The finish registration step executes this statement:

```
ALTER USER user factor FINISH REGISTRATION SET CHALLENGE_RESPONSE AS 'auth_string'
```

The statement completes the registration and sends the following information to the server as part of the auth_string: authenticator data, an optional attestation certificate in X.509 format, and a

signature.

The initiate and registration steps must be performed in a single connection, as the challenge received by the client during the initiate step is saved to the client connection handler.

Registration would fail if the registration step was performed by a different connection. The --fido-register-factor option executes both the initiate and registration steps, which avoids the failure scenario described above and prevents having to execute the ALTER USER initiate and registration statements manually.

The --fido-register-factor option is only available for the mysql client and MySQL Shell. Other MySQL client programs do not support it.

For related information, see the section called ?Using FIDO Authentication?.

? --force, -f Continue even if an SQL error occurs.

? --get-server-public-key Request from the server the public key required for RSA key pair-based password exchange. This option applies to clients that authenticate with the caching_sha2_password authentication plugin. For that plugin, the server does not send the public key unless requested. This option is ignored for accounts that do not authenticate with that plugin. It is also ignored if RSA-based password exchange is not used, as is the case when the client connects to the server using a secure connection.

If --server-public-key-path=file_name is given and specifies a valid public key file, it takes precedence over --get-server-public-key.

For information about the caching_sha2_password plugin, see Section 6.4.1.2, ?Caching SHA-2 Pluggable Authentication?.

? --histignore A list of one or more colon-separated patterns specifying statements to ignore for logging purposes. These patterns are added to the default pattern list ("*IDENTIFIED*:*PASSWORD*"). The value specified for this option affects logging of statements written to the history file, and to

syslog if the --syslog option is given. For more information, see the section called ?MYSQL CLIENT LOGGING?.

? --host=host_name, -h host_name Connect to the MySQL server on the given host.

The --dns-srv-name option takes precedence over the --host option if both are given. --dns-srv-name causes connection establishment to use the mysql_real_connect_dns_srv() C API function rather than mysql_real_connect(). However, if the connect command is subsequently used at runtime and specifies a host name argument, that host name takes precedence over any --dns-srv-name option given at mysql startup to specify a DNS SRV record.

? --html, -H Produce HTML output.

? --ignore-spaces, -i Ignore spaces after function names. The effect of this is described in the discussion for the IGNORE_SPACE SQL mode (see Section 5.1.11, ?Server SQL Modes?).

? --init-command=str SQL statement to execute after connecting to the server. If auto-reconnect is enabled, the statement is executed again after reconnection occurs.

? --line-numbers Write line numbers for errors. Disable this with --skip-line-numbers.

? --load-data-local-dir=dir_name This option affects the client-side LOCAL capability for LOAD DATA operations. It specifies the directory in which files named in LOAD DATA LOCAL statements must be located. The effect of --load-data-local-dir depends on whether LOCAL data loading is enabled or disabled:

  ? If LOCAL data loading is enabled, either by default in the MySQL client library or by specifying --local-infile[=1], the --load-data-local-dir option is ignored.

  ? If LOCAL data loading is disabled, either by default in the MySQL client library or by specifying --local-infile=0, the --load-data-local-dir option applies.

When --load-data-local-dir applies, the option value designates the directory in which local data files must be located. Comparison of

the directory path name and the path name of files to be loaded is case-sensitive regardless of the case sensitivity of the underlying file system. If the option value is the empty string, it names no directory, with the result that no files are permitted for local data loading.

For example, to explicitly disable local data loading except for files located in the /my/local/data directory, invoke mysql like this:

```
mysql --local-infile=0 --load-data-local-dir=/my/local/data
```

When both --local-infile and --load-data-local-dir are given, the order in which they are given does not matter.

Successful use of LOCAL load operations within mysql also requires that the server permits local loading; see Section 6.1.6, ?Security Considerations for LOAD DATA LOCAL?

The --load-data-local-dir option was added in MySQL 8.0.21.

? --local-infile[={0|1}] By default, LOCAL capability for LOAD DATA is determined by the default compiled into the MySQL client library. To enable or disable LOCAL data loading explicitly, use the --local-infile option. When given with no value, the option enables LOCAL data loading. When given as --local-infile=0 or --local-infile=1, the option disables or enables LOCAL data loading.

If LOCAL capability is disabled, the --load-data-local-dir option can be used to permit restricted local loading of files located in a designated directory.

Successful use of LOCAL load operations within mysql also requires that the server permits local loading; see Section 6.1.6, ?Security Considerations for LOAD DATA LOCAL?

? --login-path=name Read options from the named login path in the .mylogin.cnf login path file. A ?login path? is an option group containing options that specify which MySQL server to connect to and which account to authenticate as. To create or modify a login path file, use the mysql_config_editor utility. See

mysql_config_editor(1).

For additional information about this and other option-file options, see Section 4.2.2.3, ?Command-Line Options that Affect Option-File Handling?.

? --max-allowed-packet=value The maximum size of the buffer for client/server communication. The default is 16MB, the maximum is 1GB.

? --max-join-size=value The automatic limit for rows in a join when using --safe-updates. (Default value is 1,000,000.)

? --named-commands, -G Enable named mysql commands. Long-format commands are permitted, not just short-format commands. For example, quit and \q both are recognized. Use --skip-named-commands to disable named commands. See the section called ?MYSQL CLIENT COMMANDS?.

? --net-buffer-length=value The buffer size for TCP/IP and socket communication. (Default value is 16KB.)

? --network-namespace=name The network namespace to use for TCP/IP connections. If omitted, the connection uses the default (global) namespace. For information about network namespaces, see Section 5.1.14, ?Network Namespace Support?.
This option was added in MySQL 8.0.22. It is available only on platforms that implement network namespace support.

? --no-auto-rehash, -A This has the same effect as --skip-auto-rehash. See the description for --auto-rehash.

? --no-beep, -b Do not beep when errors occur.

? --no-defaults Do not read any option files. If program startup fails due to reading unknown options from an option file, --no-defaults can be used to prevent them from being read.
The exception is that the .mylogin.cnf file is read in all cases, if it exists. This permits passwords to be specified in a safer way than on the command line even when --no-defaults is used. To create .mylogin.cnf, use the mysql_config_editor utility. See mysql_config_editor(1).

For additional information about this and other option-file options, see Section 4.2.2.3, ?Command-Line Options that Affect Option-File Handling?.

? --one-database, -o Ignore statements except those that occur while the default database is the one named on the command line. This option is rudimentary and should be used with care. Statement filtering is based only on USE statements.

Initially, mysql executes statements in the input because specifying a database db_name on the command line is equivalent to inserting USE db_name at the beginning of the input. Then, for each USE statement encountered, mysql accepts or rejects following statements depending on whether the database named is the one on the command line. The content of the statements is immaterial.

Suppose that mysql is invoked to process this set of statements:

```
DELETE FROM db2.t2;
USE db2;
DROP TABLE db1.t1;
CREATE TABLE db1.t1 (i INT);
USE db1;
INSERT INTO t1 (i) VALUES(1);
CREATE TABLE db2.t1 (j INT);
```

If the command line is mysql --force --one-database db1, mysql handles the input as follows:

? The DELETE statement is executed because the default database is db1, even though the statement names a table in a different database.

? The DROP TABLE and CREATE TABLE statements are not executed because the default database is not db1, even though the statements name a table in db1.

? The INSERT and CREATE TABLE statements are executed because the default database is db1, even though the CREATE TABLE statement names a table in a different database.

? --pager[=command] Use the given command for paging query output. If

the command is omitted, the default pager is the value of your PAGER environment variable. Valid pagers are less, more, cat [> filename], and so forth. This option works only on Unix and only in interactive mode. To disable paging, use --skip-pager. the section called ?MYSQL CLIENT COMMANDS?, discusses output paging further.

? --password[=password], -p[password] The password of the MySQL account used for connecting to the server. The password value is optional. If not given, mysql prompts for one. If given, there must be no space between --password= or -p and the password following it. If no password option is specified, the default is to send no password.

Specifying a password on the command line should be considered insecure. To avoid giving the password on the command line, use an option file. See Section 6.1.2.1, ?End-User Guidelines for Password Security?.

To explicitly specify that there is no password and that mysql should not prompt for one, use the --skip-password option.

? --password1[=pass_val] The password for multifactor authentication factor 1 of the MySQL account used for connecting to the server. The password value is optional. If not given, mysql prompts for one. If given, there must be no space between --password1= and the password following it. If no password option is specified, the default is to send no password.

Specifying a password on the command line should be considered insecure. To avoid giving the password on the command line, use an option file. See Section 6.1.2.1, ?End-User Guidelines for Password Security?.

To explicitly specify that there is no password and that mysql should not prompt for one, use the --skip-password1 option.

--password1 and --password are synonymous, as are --skip-password1 and --skip-password.

? --password2[=pass_val] The password for multifactor authentication factor 2 of the MySQL account used for connecting to the server.

The semantics of this option are similar to the semantics for --password1; see the description of that option for details.

? --password3[=pass_val] The password for multifactor authentication factor 3 of the MySQL account used for connecting to the server. The semantics of this option are similar to the semantics for --password1; see the description of that option for details.

? --pipe, -W On Windows, connect to the server using a named pipe. This option applies only if the server was started with the named_pipe system variable enabled to support named-pipe connections. In addition, the user making the connection must be a member of the Windows group specified by the named_pipe_full_access_group system variable.

? --plugin-authentication-kerberos-client-mode=value On Windows, the authentication_kerberos_client authentication plugin supports this plugin option. It provides two possible values that the client user can set at runtime: SSPI and GSSAPI.
The default value for the client-side plugin option uses Security Support Provider Interface (SSPI), which is capable of acquiring credentials from the Windows in-memory cache. Alternatively, the client user can select a mode that supports Generic Security Service Application Program Interface (GSSAPI) through the MIT Kerberos library on Windows. GSSAPI is capable of acquiring cached credentials previously generated by using the kinit command.
For more information, see Commands for Windows Clients in GSSAPI Mode.

? --plugin-dir=dir_name The directory in which to look for plugins. Specify this option if the --default-auth option is used to specify an authentication plugin but mysql does not find it. See Section 6.2.17, ?Pluggable Authentication?.

? --port=port_num, -P port_num For TCP/IP connections, the port number to use.

? --print-defaults Print the program name and all options that it gets from option files.

For additional information about this and other option-file options, see Section 4.2.2.3, ?Command-Line Options that Affect Option-File Handling?.

? --prompt=format_str Set the prompt to the specified format. The default is mysql>. The special sequences that the prompt can contain are described in the section called ?MYSQL CLIENT COMMANDS?.

? --protocol={TCP|SOCKET|PIPE|MEMORY} The transport protocol to use for connecting to the server. It is useful when the other connection parameters normally result in use of a protocol other than the one you want. For details on the permissible values, see Section 4.2.7, ?Connection Transport Protocols?.

? --quick, -q Do not cache each query result, print each row as it is received. This may slow down the server if the output is suspended. With this option, mysql does not use the history file.

? --raw, -r For tabular output, the ?boxing? around columns enables one column value to be distinguished from another. For nontabular output (such as is produced in batch mode or when the --batch or --silent option is given), special characters are escaped in the output so they can be identified easily. Newline, tab, NUL, and backslash are written as \n, \t, \0, and \\. The --raw option disables this character escaping.

The following example demonstrates tabular versus nontabular output and the use of raw mode to disable escaping:

```
% mysql
mysql> SELECT CHAR(92);
+----------+
| CHAR(92) |
+----------+
| \        |
+----------+
% mysql -s
mysql> SELECT CHAR(92);
```

CHAR(92)

\\

% mysql -s -r

mysql> SELECT CHAR(92);

CHAR(92)

\

? --reconnect If the connection to the server is lost, automatically try to reconnect. A single reconnect attempt is made each time the connection is lost. To suppress reconnection behavior, use --skip-reconnect.

? --safe-updates, --i-am-a-dummy, -U If this option is enabled, UPDATE and DELETE statements that do not use a key in the WHERE clause or a LIMIT clause produce an error. In addition, restrictions are placed on SELECT statements that produce (or are estimated to produce) very large result sets. If you have set this option in an option file, you can use --skip-safe-updates on the command line to override it. For more information about this option, see Using Safe-Updates Mode (--safe-updates).

? --select-limit=value The automatic limit for SELECT statements when using --safe-updates. (Default value is 1,000.)

? --server-public-key-path=file_name The path name to a file in PEM format containing a client-side copy of the public key required by the server for RSA key pair-based password exchange. This option applies to clients that authenticate with the sha256_password or caching_sha2_password authentication plugin. This option is ignored for accounts that do not authenticate with one of those plugins. It is also ignored if RSA-based password exchange is not used, as is the case when the client connects to the server using a secure connection.

If --server-public-key-path=file_name is given and specifies a valid public key file, it takes precedence over --get-server-public-key.

For sha256_password, this option applies only if MySQL was built

using OpenSSL.

For information about the sha256_password and caching_sha2_password plugins, see Section 6.4.1.3, ?SHA-256 Pluggable Authentication?, and Section 6.4.1.2, ?Caching SHA-2 Pluggable Authentication?.

? --shared-memory-base-name=name On Windows, the shared-memory name to use for connections made using shared memory to a local server. The default value is MYSQL. The shared-memory name is case-sensitive.

This option applies only if the server was started with the shared_memory system variable enabled to support shared-memory connections.

? --show-warnings Cause warnings to be shown after each statement if there are any. This option applies to interactive and batch mode.

? --sigint-ignore Ignore SIGINT signals (typically the result of typing Control+C).

Without this option, typing Control+C interrupts the current statement if there is one, or cancels any partial input line otherwise.

? --silent, -s Silent mode. Produce less output. This option can be given multiple times to produce less and less output.

This option results in nontabular output format and escaping of special characters. Escaping may be disabled by using raw mode; see the description for the --raw option.

? --skip-column-names, -N Do not write column names in results.

? --skip-line-numbers, -L Do not write line numbers for errors.

Useful when you want to compare result files that include error messages.

? --socket=path, -S path For connections to localhost, the Unix socket file to use, or, on Windows, the name of the named pipe to use.

On Windows, this option applies only if the server was started with the named_pipe system variable enabled to support named-pipe connections. In addition, the user making the connection must be a

member of the Windows group specified by the named_pipe_full_access_group system variable.

? --ssl* Options that begin with --ssl specify whether to connect to the server using encryption and indicate where to find SSL keys and certificates. See the section called ?Command Options for Encrypted Connections?.

? --ssl-fips-mode={OFF|ON|STRICT} Controls whether to enable FIPS mode on the client side. The --ssl-fips-mode option differs from other --ssl-xxx options in that it is not used to establish encrypted connections, but rather to affect which cryptographic operations to permit. See Section 6.8, ?FIPS Support?.

These --ssl-fips-mode values are permitted:

? OFF: Disable FIPS mode.

? ON: Enable FIPS mode.

? STRICT: Enable ?strict? FIPS mode.

Note

If the OpenSSL FIPS Object Module is not available, the only permitted value for --ssl-fips-mode is OFF. In this case, setting --ssl-fips-mode to ON or STRICT causes the client to produce a warning at startup and to operate in non-FIPS mode.

As of MySQL 8.0.34, this option is deprecated. Expect it to be removed in a future version of MySQL.

? --syslog, -j This option causes mysql to send interactive statements to the system logging facility. On Unix, this is syslog; on Windows, it is the Windows Event Log. The destination where logged messages appear is system dependent. On Linux, the destination is often the /var/log/messages file.

Here is a sample of output generated on Linux by using --syslog. This output is formatted for readability; each logged message actually takes a single line.

```
Mar  7 12:39:25 myhost MysqlClient[20824]:
  SYSTEM_USER:'oscar', MYSQL_USER:'my_oscar', CONNECTION_ID:23,
  DB_SERVER:'127.0.0.1', DB:'--', QUERY:'USE test;'
```

Mar 7 12:39:28 myhost MysqlClient[20824]:
SYSTEM_USER:'oscar', MYSQL_USER:'my_oscar', CONNECTION_ID:23,
DB_SERVER:'127.0.0.1', DB:'test', QUERY:'SHOW TABLES;'

For more information, see the section called ?MYSQL CLIENT LOGGING?.

? --table, -t Display output in table format. This is the default for interactive use, but can be used to produce table output in batch mode.

? --tee=file_name Append a copy of output to the given file. This option works only in interactive mode. the section called ?MYSQL CLIENT COMMANDS?, discusses tee files further.

? --tls-ciphersuites=ciphersuite_list The permissible ciphersuites for encrypted connections that use TLSv1.3. The value is a list of one or more colon-separated ciphersuite names. The ciphersuites that can be named for this option depend on the SSL library used to compile MySQL. For details, see Section 6.3.2, ?Encrypted Connection TLS Protocols and Ciphers?.
This option was added in MySQL 8.0.16.

? --tls-version=protocol_list The permissible TLS protocols for encrypted connections. The value is a list of one or more comma-separated protocol names. The protocols that can be named for this option depend on the SSL library used to compile MySQL. For details, see Section 6.3.2, ?Encrypted Connection TLS Protocols and Ciphers?.

? --unbuffered, -n Flush the buffer after each query.

? --user=user_name, -u user_name The user name of the MySQL account to use for connecting to the server.

? --verbose, -v Verbose mode. Produce more output about what the program does. This option can be given multiple times to produce more and more output. (For example, -v -v -v produces table output format even in batch mode.)

? --version, -V Display version information and exit.

? --vertical, -E Print query output rows vertically (one line per

column value). Without this option, you can specify vertical output for individual statements by terminating them with \G.

? --wait, -w If the connection cannot be established, wait and retry instead of aborting.

? --xml, -X Produce XML output.

<field name="column_name">NULL</field>

The output when --xml is used with mysql matches that of mysqldump --xml. See mysqldump(1), for details.

The XML output also uses an XML namespace, as shown here:

$> mysql --xml -uroot -e "SHOW VARIABLES LIKE 'version%'"

<?xml version="1.0"?>

<resultset statement="SHOW VARIABLES LIKE 'version%'" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<row>

<field name="Variable_name">version</field>

<field name="Value">5.0.40-debug</field>

</row>

<row>

<field name="Variable_name">version_comment</field>

<field name="Value">Source distribution</field>

</row>

<row>

<field name="Variable_name">version_compile_machine</field>

<field name="Value">i686</field>

</row>

<row>

<field name="Variable_name">version_compile_os</field>

<field name="Value">suse-linux-gnu</field>

</row>

</resultset>

? --zstd-compression-level=level The compression level to use for connections to the server that use the zstd compression algorithm.

The permitted levels are from 1 to 22, with larger values

indicating increasing levels of compression. The default zstd
compression level is 3. The compression level setting has no effect
on connections that do not use zstd compression.
For more information, see Section 4.2.8, ?Connection Compression
Control?.
This option was added in MySQL 8.0.18.

MYSQL CLIENT COMMANDS

mysql sends each SQL statement that you issue to the server to be
executed. There is also a set of commands that mysql itself interprets.
For a list of these commands, type help or \h at the mysql> prompt:

```
mysql> help
List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
?         (\?) Synonym for `help'.
clear     (\c) Clear the current input statement.
connect   (\r) Reconnect to the server. Optional arguments are db and host.
delimiter (\d) Set statement delimiter.
edit      (\e) Edit command with $EDITOR.
ego       (\G) Send command to mysql server, display result vertically.
exit      (\q) Exit mysql. Same as quit.
go        (\g) Send command to mysql server.
help      (\h) Display this help.
nopager   (\n) Disable pager, print to stdout.
notee     (\t) Don't write into outfile.
pager     (\P) Set PAGER [to_pager]. Print the query results via PAGER.
print     (\p) Print current command.
prompt    (\R) Change your mysql prompt.
quit      (\q) Quit mysql.
rehash    (\#) Rebuild completion hash.
source    (\.) Execute an SQL script file. Takes a file name as an argument.
status    (\s) Get status information from the server.
system    (\!) Execute a system shell command.
tee       (\T) Set outfile [to_outfile]. Append everything into given
```

outfile.

use (\u) Use another database. Takes database name as argument.

charset (\C) Switch to another charset. Might be needed for processing binlog with multi-byte charsets.

warnings (\W) Show warnings after every statement.

nowarning (\w) Don't show warnings after every statement.

resetconnection(\x) Clean session context.

query_attributes Sets string parameters (name1 value1 name2 value2 ...) for the next query to pick up.

ssl_session_data_print Serializes the current SSL session data to stdout or file.

For server side help, type 'help contents'

If mysql is invoked with the --binary-mode option, all mysql commands are disabled except charset and delimiter in noninteractive mode (for input piped to mysql or loaded using the source command).

Each command has both a long and short form. The long form is not case-sensitive; the short form is. The long form can be followed by an optional semicolon terminator, but the short form should not.

The use of short-form commands within multiple-line /* ... */ comments is not supported. Short-form commands do work within single-line /*! ... */ version comments, as do /*+ ... */ optimizer-hint comments, which are stored in object definitions. If there is a concern that optimizer-hint comments may be stored in object definitions so that dump files when reloaded with mysql would result in execution of such commands, either invoke mysql with the --binary-mode option or use a reload client other than mysql.

? help [arg], \h [arg], \? [arg], ? [arg]

Display a help message listing the available mysql commands.

If you provide an argument to the help command, mysql uses it as a search string to access server-side help from the contents of the MySQL Reference Manual. For more information, see the section called ?MYSQL CLIENT SERVER-SIDE HELP?.

? charset charset_name, \C charset_name

Change the default character set and issue a SET NAMES statement. This enables the character set to remain synchronized on the client and server if mysql is run with auto-reconnect enabled (which is not recommended), because the specified character set is used for reconnects.

? clear, \c

Clear the current input. Use this if you change your mind about executing the statement that you are entering.

? connect [db_name [host_name]], \r [db_name [host_name]]

Reconnect to the server. The optional database name and host name arguments may be given to specify the default database or the host where the server is running. If omitted, the current values are used.

If the connect command specifies a host name argument, that host takes precedence over any --dns-srv-name option given at mysql startup to specify a DNS SRV record.

? delimiter str, \d str

Change the string that mysql interprets as the separator between SQL statements. The default is the semicolon character (;).

The delimiter string can be specified as an unquoted or quoted argument on the delimiter command line. Quoting can be done with either single quote ('), double quote ("), or backtick (`) characters. To include a quote within a quoted string, either quote the string with a different quote character or escape the quote with a backslash (\) character. Backslash should be avoided outside of quoted strings because it is the escape character for MySQL. For an unquoted argument, the delimiter is read up to the first space or end of line. For a quoted argument, the delimiter is read up to the matching quote on the line.

mysql interprets instances of the delimiter string as a statement delimiter anywhere it occurs, except within quoted strings. Be careful about defining a delimiter that might occur within other words. For example, if you define the delimiter as X, it is not

possible to use the word INDEX in statements. mysql interprets this as INDE followed by the delimiter X.

When the delimiter recognized by mysql is set to something other than the default of ;, instances of that character are sent to the server without interpretation. However, the server itself still interprets ; as a statement delimiter and processes statements accordingly. This behavior on the server side comes into play for multiple-statement execution (see Multiple Statement Execution Support[3]), and for parsing the body of stored procedures and functions, triggers, and events (see Section 25.1, ?Defining Stored Programs?).

? edit, \e

Edit the current input statement. mysql checks the values of the EDITOR and VISUAL environment variables to determine which editor to use. The default editor is vi if neither variable is set.

The edit command works only in Unix.

? ego, \G

Send the current statement to the server to be executed and display the result using vertical format.

? exit, \q

Exit mysql.

? go, \g

Send the current statement to the server to be executed.

? nopager, \n

Disable output paging. See the description for pager.

The nopager command works only in Unix.

? notee, \t

Disable output copying to the tee file. See the description for tee.

? nowarning, \w

Disable display of warnings after each statement.

? pager [command], \P [command]

Enable output paging. By using the --pager option when you invoke

mysql, it is possible to browse or search query results in interactive mode with Unix programs such as less, more, or any other similar program. If you specify no value for the option, mysql checks the value of the PAGER environment variable and sets the pager to that. Pager functionality works only in interactive mode.

Output paging can be enabled interactively with the pager command and disabled with nopager. The command takes an optional argument; if given, the paging program is set to that. With no argument, the pager is set to the pager that was set on the command line, or stdout if no pager was specified.

Output paging works only in Unix because it uses the popen() function, which does not exist on Windows. For Windows, the tee option can be used instead to save query output, although it is not as convenient as pager for browsing output in some situations.

? print, \p

Print the current input statement without executing it.

? prompt [str], \R [str]

Reconfigure the mysql prompt to the given string. The special character sequences that can be used in the prompt are described later in this section.

If you specify the prompt command with no argument, mysql resets the prompt to the default of mysql>.

? query_attributes name value [name value ...]

Define query attributes that apply to the next query sent to the server. For discussion of the purpose and use of query attributes, see Section 9.6, ?Query Attributes?.

The query_attributes command follows these rules:

- ? The format and quoting rules for attribute names and values are the same as for the delimiter command.
- ? The command permits up to 32 attribute name/value pairs. Names and values may be up to 1024 characters long. If a name is given without a value, an error occurs.

? If multiple query_attributes commands are issued prior to query execution, only the last command applies. After sending the query, mysql clears the attribute set.

? If multiple attributes are defined with the same name, attempts to retrieve the attribute value have an undefined result.

? An attribute defined with an empty name cannot be retrieved by name.

? If a reconnect occurs while mysql executes the query, mysql restores the attributes after reconnecting so the query can be executed again with the same attributes.

? quit, \q

Exit mysql.

? rehash, \#

Rebuild the completion hash that enables database, table, and column name completion while you are entering statements. (See the description for the --auto-rehash option.)

? resetconnection, \x

Reset the connection to clear the session state. This includes clearing any current query attributes defined using the query_attributes command.

Resetting a connection has effects similar to mysql_change_user() or an auto-reconnect except that the connection is not closed and reopened, and re-authentication is not done. See mysql_change_user()[4], and Automatic Reconnection Control[5].

This example shows how resetconnection clears a value maintained in the session state:

```
mysql> SELECT LAST_INSERT_ID(3);
+-------------------+
| LAST_INSERT_ID(3) |
+-------------------+
|                 3 |
+-------------------+
mysql> SELECT LAST_INSERT_ID();
```

```
+------------------+
| LAST_INSERT_ID() |
+------------------+
|                3 |
+------------------+
mysql> resetconnection;
mysql> SELECT LAST_INSERT_ID();
+------------------+
| LAST_INSERT_ID() |
+------------------+
|                0 |
+------------------+
```

? source file_name, \. file_name

Read the named file and executes the statements contained therein.
On Windows, specify path name separators as / or \\.
Quote characters are taken as part of the file name itself. For
best results, the name should not include space characters.

? ssl_session_data_print [file_name]

Fetches, serializes, and optionally stores the session data of a
successful connection. The optional file name and arguments may be
given to specify the file to store serialized session data. If
omitted, the session data is printed to stdout.
If the MySQL session is configured for reuse, session data from the
file is deserialized and supplied to the connect command to
reconnect. When the session is reused successfully, the status
command contains a row showing SSL session reused: true while the
client remains reconnected to the server.

? status, \s

Provide status information about the connection and the server you
are using. If you are running with --safe-updates enabled, status
also prints the values for the mysql variables that affect your
queries.

? system command, \! command

Execute the given command using your default command interpreter.

Prior to MySQL 8.0.19, the system command works only in Unix. As of 8.0.19, it also works on Windows.

? tee [file_name], \T [file_name]

By using the --tee option when you invoke mysql, you can log statements and their output. All the data displayed on the screen is appended into a given file. This can be very useful for debugging purposes also. mysql flushes results to the file after each statement, just before it prints its next prompt. Tee functionality works only in interactive mode.

You can enable this feature interactively with the tee command. Without a parameter, the previous file is used. The tee file can be disabled with the notee command. Executing tee again re-enables logging.

? use db_name, \u db_name

Use db_name as the default database.

? warnings, \W

Enable display of warnings after each statement (if there are any).

Here are a few tips about the pager command:

? You can use it to write to a file and the results go only to the file:

```
mysql> pager cat > /tmp/log.txt
```

You can also pass any options for the program that you want to use as your pager:

```
mysql> pager less -n -i -S
```

? In the preceding example, note the -S option. You may find it very useful for browsing wide query results. Sometimes a very wide result set is difficult to read on the screen. The -S option to less can make the result set much more readable because you can scroll it horizontally using the left-arrow and right-arrow keys.

You can also use -S interactively within less to switch the horizontal-browse mode on and off. For more information, read the less manual page:

man less

? The -F and -X options may be used with less to cause it to exit if output fits on one screen, which is convenient when no scrolling is necessary:

mysql> pager less -n -i -S -F -X

? You can specify very complex pager commands for handling query output:

mysql> pager cat | tee /dr1/tmp/res.txt \
| tee /dr2/tmp/res2.txt | less -n -i -S

In this example, the command would send query results to two files in two different directories on two different file systems mounted on /dr1 and /dr2, yet still display the results onscreen using less.

You can also combine the tee and pager functions. Have a tee file enabled and pager set to less, and you are able to browse the results using the less program and still have everything appended into a file the same time. The difference between the Unix tee used with the pager command and the mysql built-in tee command is that the built-in tee works even if you do not have the Unix tee available. The built-in tee also logs everything that is printed on the screen, whereas the Unix tee used with pager does not log quite that much. Additionally, tee file logging can be turned on and off interactively from within mysql. This is useful when you want to log some queries to a file, but not others.

The prompt command reconfigures the default mysql> prompt. The string for defining the prompt can contain the following special sequences.

r

.br

.br

72

??

?Option ? Description ?

??

? ? The current connection ?

? ? identifier ?

??

? ? A counter that increments ?

? ? for each statement you ?

? ? issue ?

??

? ? The full current date ?

??

? ? The default database ?

??

? ? The server host ?

??

? ? The current delimiter ?

??

? ? Minutes of the current ?

? ? time ?

??

? ? A newline character ?

??

? ? The current month in ?

? ? three-letter format (Jan, ?

? ? Feb, ...) ?

??

? ? The current month in ?

? ? numeric format ?

??

?P ? am/pm ?

??

? ? The current TCP/IP port or ?

? ? socket file ?

??

? ? The current time, in ?

? ? 24-hour military time ?

? ? (0?23) ?

??

? ? The current time, standard ?

? ? 12-hour time (1?12) ?

??

? ? Semicolon ?

??

? ? Seconds of the current ?

? ? time ?

??

?T ? Print an asterisk (*) if ?

? ? the current session is ?

? ? inside a ?

? ? transaction block (from ?

? ? MySQL 8.0.28) ?

??

? ? A tab character ?

??

?U ? ?

? ? Your full ?

? ? user_name@host_name ?

? ? account name ?

??

? ? Your user name ?

??

? ? The server version ?

??

? ? The current day of the ?

? ? week in three-letter ?

? ? format (Mon, Tue, ...) ?

??

? ? The current year, four ?

? ? digits ?

??

?y ? The current year, two ?

? ? digits ?

??

?_ ? A space ?

??

?\ ? A space (a space follows ?

? ? the backslash) ?

??

?? ? Single quote ?

??

? ? Double quote ?

??

?T}:T{ A literal backslash ? ?

?character ? ?

??

?\flx ? ?

? ? x, for any ?x? not ?

? ? listed above ?

??

You can set the prompt in several ways:

? Use an environment variable. You can set the MYSQL_PS1 environment variable to a prompt string. For example:

```
export MYSQL_PS1="(\u@\h) [\d]> "
```

? Use a command-line option. You can set the --prompt option on the command line to mysql. For example:

```
$> mysql --prompt="(\u@\h) [\d]> "
(user@host) [database]>
```

? Use an option file. You can set the prompt option in the [mysql] group of any MySQL option file, such as /etc/my.cnf or the .my.cnf file in your home directory. For example:

```
[mysql]
```

prompt=(\\u@\\h) [\\d]>_

In this example, note that the backslashes are doubled. If you set the prompt using the prompt option in an option file, it is advisable to double the backslashes when using the special prompt options. There is some overlap in the set of permissible prompt options and the set of special escape sequences that are recognized in option files. (The rules for escape sequences in option files are listed in Section 4.2.2.2, ?Using Option Files?.) The overlap may cause you problems if you use single backslashes. For example, \s is interpreted as a space rather than as the current seconds value. The following example shows how to define a prompt within an option file to include the current time in hh:mm:ss> format:

[mysql]

prompt="\\r:\\m:\\s> "

? Set the prompt interactively. You can change your prompt interactively by using the prompt (or \R) command. For example:

mysql> prompt (\u@\h) [\d]>_

PROMPT set to '(\u@\h) [\d]>_'

(user@host) [database]>

(user@host) [database]> prompt

Returning to default PROMPT of mysql>

mysql>

MYSQL CLIENT LOGGING

The mysql client can do these types of logging for statements executed interactively:

? On Unix, mysql writes the statements to a history file. By default, this file is named .mysql_history in your home directory. To specify a different file, set the value of the MYSQL_HISTFILE environment variable.

? On all platforms, if the --syslog option is given, mysql writes the statements to the system logging facility. On Unix, this is syslog; on Windows, it is the Windows Event Log. The destination where logged messages appear is system dependent. On Linux, the

destination is often the /var/log/messages file.

The following discussion describes characteristics that apply to all logging types and provides information specific to each logging type.

? How Logging Occurs

? Controlling the History File

? syslog Logging Characteristics

How Logging Occurs

For each enabled logging destination, statement logging occurs as follows:

? Statements are logged only when executed interactively. Statements are noninteractive, for example, when read from a file or a pipe. It is also possible to suppress statement logging by using the --batch or --execute option.

? Statements are ignored and not logged if they match any pattern in the ?ignore? list. This list is described later.

? mysql logs each nonignored, nonempty statement line individually.

? If a nonignored statement spans multiple lines (not including the terminating delimiter), mysql concatenates the lines to form the complete statement, maps newlines to spaces, and logs the result, plus a delimiter.

Consequently, an input statement that spans multiple lines can be logged twice. Consider this input:

```
mysql> SELECT
    -> 'Today is'
    -> ,
    -> CURDATE()
    -> ;
```

In this case, mysql logs the ?SELECT?, ?'Today is'?, ?,?, ?CURDATE()?, and ?;? lines as it reads them. It also logs the complete statement, after mapping SELECT\n'Today is'\n,\nCURDATE() to SELECT 'Today is' , CURDATE(), plus a delimiter. Thus, these lines appear in logged output:

```
SELECT
'Today is'
```

,

CURDATE()

;

SELECT 'Today is' , CURDATE();

mysql ignores for logging purposes statements that match any pattern in the ?ignore? list. By default, the pattern list is "*IDENTIFIED*:*PASSWORD*", to ignore statements that refer to passwords. Pattern matching is not case-sensitive. Within patterns, two characters are special:

? ? matches any single character.

? * matches any sequence of zero or more characters.

To specify additional patterns, use the --histignore option or set the MYSQL_HISTIGNORE environment variable. (If both are specified, the option value takes precedence.) The value should be a list of one or more colon-separated patterns, which are appended to the default pattern list.

Patterns specified on the command line might need to be quoted or escaped to prevent your command interpreter from treating them specially. For example, to suppress logging for UPDATE and DELETE statements in addition to statements that refer to passwords, invoke mysql like this:

mysql --histignore="*UPDATE*:*DELETE*"

Controlling the History File

The .mysql_history file should be protected with a restrictive access mode because sensitive information might be written to it, such as the text of SQL statements that contain passwords. See Section 6.1.2.1, ?End-User Guidelines for Password Security?. Statements in the file are accessible from the mysql client when the up-arrow key is used to recall the history. See Disabling Interactive History.

If you do not want to maintain a history file, first remove .mysql_history if it exists. Then use either of the following techniques to prevent it from being created again:

? Set the MYSQL_HISTFILE environment variable to /dev/null. To cause

this setting to take effect each time you log in, put it in one of your shell's startup files.

? Create .mysql_history as a symbolic link to /dev/null; this need be done only once:

ln -s /dev/null $HOME/.mysql_history

syslog Logging Characteristics

If the --syslog option is given, mysql writes interactive statements to the system logging facility. Message logging has the following characteristics.

Logging occurs at the ?information? level. This corresponds to the LOG_INFO priority for syslog on Unix/Linux syslog capability and to EVENTLOG_INFORMATION_TYPE for the Windows Event Log. Consult your system documentation for configuration of your logging capability.

Message size is limited to 1024 bytes.

Messages consist of the identifier MysqlClient followed by these values:

? SYSTEM_USER

The operating system user name (login name) or -- if the user is unknown.

? MYSQL_USER

The MySQL user name (specified with the --user option) or -- if the user is unknown.

? CONNECTION_ID:

The client connection identifier. This is the same as the CONNECTION_ID() function value within the session.

? DB_SERVER

The server host or -- if the host is unknown.

? DB

The default database or -- if no database has been selected.

? QUERY

The text of the logged statement.

Here is a sample of output generated on Linux by using --syslog. This output is formatted for readability; each logged message actually takes

a single line.

Mar 7 12:39:25 myhost MysqlClient[20824]:

SYSTEM_USER:'oscar', MYSQL_USER:'my_oscar', CONNECTION_ID:23,

DB_SERVER:'127.0.0.1', DB:'--', QUERY:'USE test;'

Mar 7 12:39:28 myhost MysqlClient[20824]:

SYSTEM_USER:'oscar', MYSQL_USER:'my_oscar', CONNECTION_ID:23,

DB_SERVER:'127.0.0.1', DB:'test', QUERY:'SHOW TABLES;'

MYSQL CLIENT SERVER-SIDE HELP

mysql> help search_string

If you provide an argument to the help command, mysql uses it as a search string to access server-side help from the contents of the MySQL Reference Manual. The proper operation of this command requires that the help tables in the mysql database be initialized with help topic information (see Section 5.1.17, ?Server-Side Help Support?).

If there is no match for the search string, the search fails:

mysql> help me

Nothing found

Please try to run 'help contents' for a list of all accessible topics

Use help contents to see a list of the help categories:

mysql> help contents

You asked for help about help category: "Contents"

For more information, type 'help <item>', where <item> is one of the following categories:

Account Management

Administration

Data Definition

Data Manipulation

Data Types

Functions

Functions and Modifiers for Use with GROUP BY

Geographic Features

Language Structure

Plugins

Storage Engines

Stored Routines

Table Maintenance

Transactions

Triggers

If the search string matches multiple items, mysql shows a list of matching topics:

```
mysql> help logs
Many help items for your request exist.
To make a more specific request, please type 'help <item>',
where <item> is one of the following topics:
   SHOW
   SHOW BINARY LOGS
   SHOW ENGINE
   SHOW LOGS
```

Use a topic as the search string to see the help entry for that topic:

```
mysql> help show binary logs
Name: 'SHOW BINARY LOGS'
Description:
Syntax:
SHOW BINARY LOGS
SHOW MASTER LOGS
Lists the binary log files on the server. This statement is used as
part of the procedure described in [purge-binary-logs], that shows how
to determine which logs can be purged.
mysql> SHOW BINARY LOGS;
+---------------+-----------+-----------+
| Log_name      | File_size | Encrypted |
+---------------+-----------+-----------+
| binlog.000015 |    724935 | Yes       |
| binlog.000016 |    733481 | Yes       |
+---------------+-----------+-----------+
```

The search string can contain the wildcard characters % and _. These

have the same meaning as for pattern-matching operations performed with the LIKE operator. For example, HELP rep% returns a list of topics that begin with rep:

```
mysql> HELP rep%
Many help items for your request exist.
To make a more specific request, please type 'help <item>',
where <item> is one of the following
topics:
  REPAIR TABLE
  REPEAT FUNCTION
  REPEAT LOOP
  REPLACE
  REPLACE FUNCTION
```

EXECUTING SQL STATEMENTS FROM A TEXT FILE

The mysql client typically is used interactively, like this:

```
mysql db_name
```

However, it is also possible to put your SQL statements in a file and then tell mysql to read its input from that file. To do so, create a text file text_file that contains the statements you wish to execute. Then invoke mysql as shown here:

```
mysql db_name < text_file
```

If you place a USE db_name statement as the first statement in the file, it is unnecessary to specify the database name on the command line:

```
mysql < text_file
```

If you are already running mysql, you can execute an SQL script file using the source command or \. command:

```
mysql> source file_name
mysql> \. file_name
```

Sometimes you may want your script to display progress information to the user. For this you can insert statements like this:

```
SELECT '<info_to_display>' AS ' ';
```

The statement shown outputs <info_to_display>.

You can also invoke mysql with the --verbose option, which causes each statement to be displayed before the result that it produces.

mysql ignores Unicode byte order mark (BOM) characters at the beginning of input files. Previously, it read them and sent them to the server, resulting in a syntax error. Presence of a BOM does not cause mysql to change its default character set. To do that, invoke mysql with an option such as --default-character-set=utf8mb4.

For more information about batch mode, see Section 3.5, ?Using mysql in Batch Mode?.

MYSQL CLIENT TIPS

This section provides information about techniques for more effective use of mysql and about mysql operational behavior.

? Input-Line Editing

? Disabling Interactive History

? Unicode Support on Windows

? Displaying Query Results Vertically

? Using Safe-Updates Mode (--safe-updates)

? Disabling mysql Auto-Reconnect

? mysql Client Parser Versus Server Parser

Input-Line Editing

mysql supports input-line editing, which enables you to modify the current input line in place or recall previous input lines. For example, the left-arrow and right-arrow keys move horizontally within the current input line, and the up-arrow and down-arrow keys move up and down through the set of previously entered lines. Backspace deletes the character before the cursor and typing new characters enters them at the cursor position. To enter the line, press Enter.

On Windows, the editing key sequences are the same as supported for command editing in console windows. On Unix, the key sequences depend on the input library used to build mysql (for example, the libedit or readline library).

Documentation for the libedit and readline libraries is available online. To change the set of key sequences permitted by a given input

library, define key bindings in the library startup file. This is a file in your home directory: .editrc for libedit and .inputrc for readline.

For example, in libedit, Control+W deletes everything before the current cursor position and Control+U deletes the entire line. In readline, Control+W deletes the word before the cursor and Control+U deletes everything before the current cursor position. If mysql was built using libedit, a user who prefers the readline behavior for these two keys can put the following lines in the .editrc file (creating the file if necessary):

```
bind "^W" ed-delete-prev-word
bind "^U" vi-kill-line-prev
```

To see the current set of key bindings, temporarily put a line that says only bind at the end of .editrc. mysql shows the bindings when it starts. Disabling Interactive History

The up-arrow key enables you to recall input lines from current and previous sessions. In cases where a console is shared, this behavior may be unsuitable. mysql supports disabling the interactive history partially or fully, depending on the host platform.

On Windows, the history is stored in memory. Alt+F7 deletes all input lines stored in memory for the current history buffer. It also deletes the list of sequential numbers in front of the input lines displayed with F7 and recalled (by number) with F9. New input lines entered after you press Alt+F7 repopulate the current history buffer. Clearing the buffer does not prevent logging to the Windows Event Viewer, if the --syslog option was used to start mysql. Closing the console window also clears the current history buffer.

To disable interactive history on Unix, first delete the .mysql_history file, if it exists (previous entries are recalled otherwise). Then start mysql with the --histignore="*" option to ignore all new input lines. To re-enable the recall (and logging) behavior, restart mysql without the option.

If you prevent the .mysql_history file from being created (see

Controlling the History File) and use --histignore="*" to start the mysql client, the interactive history recall facility is disabled fully. Alternatively, if you omit the --histignore option, you can recall the input lines entered during the current session. Unicode Support on Windows

Windows provides APIs based on UTF-16LE for reading from and writing to the console; the mysql client for Windows is able to use these APIs.

The Windows installer creates an item in the MySQL menu named MySQL command line client - Unicode. This item invokes the mysql client with properties set to communicate through the console to the MySQL server using Unicode.

To take advantage of this support manually, run mysql within a console that uses a compatible Unicode font and set the default character set to a Unicode character set that is supported for communication with the server:

1. Open a console window.
2. Go to the console window properties, select the font tab, and choose Lucida Console or some other compatible Unicode font. This is necessary because console windows start by default using a DOS raster font that is inadequate for Unicode.
3. Execute mysql.exe with the --default-character-set=utf8mb4 (or utf8mb3) option. This option is necessary because utf16le is one of the character sets that cannot be used as the client character set. See the section called ?Impermissible Client Character Sets?.

With those changes, mysql uses the Windows APIs to communicate with the console using UTF-16LE, and communicate with the server using UTF-8. (The menu item mentioned previously sets the font and character set as just described.)

To avoid those steps each time you run mysql, you can create a shortcut that invokes mysql.exe. The shortcut should set the console font to Lucida Console or some other compatible Unicode font, and pass the --default-character-set=utf8mb4 (or utf8mb3) option to mysql.exe.

Alternatively, create a shortcut that only sets the console font, and

set the character set in the [mysql] group of your my.ini file:

```
[mysql]
default-character-set=utf8mb4   # or utf8mb3
```

Displaying Query Results Vertically

Some query results are much more readable when displayed vertically, instead of in the usual horizontal table format. Queries can be displayed vertically by terminating the query with \G instead of a semicolon. For example, longer text values that include newlines often are much easier to read with vertical output:

```
mysql> SELECT * FROM mails WHERE LENGTH(txt) < 300 LIMIT 300,1\G
*************************** 1. row ***************************
  msg_nro: 3068
     date: 2000-03-01 23:29:50
time_zone: +0200
mail_from: Jones
    reply: jones@example.com
  mail_to: "John Smith" <smith@example.com>
      sbj: UTF-8
      txt: >>>>> "John" == John Smith writes:
John> Hi.  I think this is a good idea.  Is anyone familiar
John> with UTF-8 or Unicode? Otherwise, I'll put this on my
John> TODO list and see what happens.
Yes, please do that.
Regards,
Jones
     file: inbox-jani-1
     hash: 190402944
1 row in set (0.09 sec)
```

Using Safe-Updates Mode (--safe-updates)

For beginners, a useful startup option is --safe-updates (or --i-am-a-dummy, which has the same effect). Safe-updates mode is helpful for cases when you might have issued an UPDATE or DELETE statement but forgotten the WHERE clause indicating which rows to

modify. Normally, such statements update or delete all rows in the table. With --safe-updates, you can modify rows only by specifying the key values that identify them, or a LIMIT clause, or both. This helps prevent accidents. Safe-updates mode also restricts SELECT statements that produce (or are estimated to produce) very large result sets.

The --safe-updates option causes mysql to execute the following statement when it connects to the MySQL server, to set the session values of the sql_safe_updates, sql_select_limit, and max_join_size system variables:

```
SET sql_safe_updates=1, sql_select_limit=1000, max_join_size=1000000;
```

The SET statement affects statement processing as follows:

? Enabling sql_safe_updates causes UPDATE and DELETE statements to produce an error if they do not specify a key constraint in the WHERE clause, or provide a LIMIT clause, or both. For example:

```
UPDATE tbl_name SET not_key_column=val WHERE key_column=val;
UPDATE tbl_name SET not_key_column=val LIMIT 1;
```

? Setting sql_select_limit to 1,000 causes the server to limit all SELECT result sets to 1,000 rows unless the statement includes a LIMIT clause.

? Setting max_join_size to 1,000,000 causes multiple-table SELECT statements to produce an error if the server estimates it must examine more than 1,000,000 row combinations.

To specify result set limits different from 1,000 and 1,000,000, you can override the defaults by using the --select-limit and --max-join-size options when you invoke mysql:

```
mysql --safe-updates --select-limit=500 --max-join-size=10000
```

It is possible for UPDATE and DELETE statements to produce an error in safe-updates mode even with a key specified in the WHERE clause, if the optimizer decides not to use the index on the key column:

? Range access on the index cannot be used if memory usage exceeds that permitted by the range_optimizer_max_mem_size system variable. The optimizer then falls back to a table scan. See the section called ?Limiting Memory Use for Range Optimization?.

? If key comparisons require type conversion, the index may not be used (see Section 8.3.1, ?How MySQL Uses Indexes?). Suppose that an indexed string column c1 is compared to a numeric value using WHERE c1 = 2222. For such comparisons, the string value is converted to a number and the operands are compared numerically (see Section 12.3, ?Type Conversion in Expression Evaluation?), preventing use of the index. If safe-updates mode is enabled, an error occurs.

As of MySQL 8.0.13, safe-updates mode also includes these behaviors:

? EXPLAIN with UPDATE and DELETE statements does not produce safe-updates errors. This enables use of EXPLAIN plus SHOW WARNINGS to see why an index is not used, which can be helpful in cases such as when a range_optimizer_max_mem_size violation or type conversion occurs and the optimizer does not use an index even though a key column was specified in the WHERE clause.

? When a safe-updates error occurs, the error message includes the first diagnostic that was produced, to provide information about the reason for failure. For example, the message may indicate that the range_optimizer_max_mem_size value was exceeded or type conversion occurred, either of which can preclude use of an index.

? For multiple-table deletes and updates, an error is produced with safe updates enabled only if any target table uses a table scan.

Disabling mysql Auto-Reconnect

If the mysql client loses its connection to the server while sending a statement, it immediately and automatically tries to reconnect once to the server and send the statement again. However, even if mysql succeeds in reconnecting, your first connection has ended and all your previous session objects and settings are lost: temporary tables, the autocommit mode, and user-defined and session variables. Also, any current transaction rolls back. This behavior may be dangerous for you, as in the following example where the server was shut down and restarted between the first and second statements without you knowing it:

```
mysql> SET @a=1;
```

```
Query OK, 0 rows affected (0.05 sec)
mysql> INSERT INTO t VALUES(@a);
ERROR 2006: MySQL server has gone away
No connection. Trying to reconnect...
Connection id:    1
Current database: test
Query OK, 1 row affected (1.30 sec)
mysql> SELECT * FROM t;
+------+
| a    |
+------+
| NULL |
+------+
1 row in set (0.05 sec)
```

The @a user variable has been lost with the connection, and after the reconnection it is undefined. If it is important to have mysql terminate with an error if the connection has been lost, you can start the mysql client with the --skip-reconnect option.

For more information about auto-reconnect and its effect on state information when a reconnection occurs, see Automatic Reconnection Control[5]. mysql Client Parser Versus Server Parser

The mysql client uses a parser on the client side that is not a duplicate of the complete parser used by the mysqld server on the server side. This can lead to differences in treatment of certain constructs. Examples:

? The server parser treats strings delimited by " characters as identifiers rather than as plain strings if the ANSI_QUOTES SQL mode is enabled.

The mysql client parser does not take the ANSI_QUOTES SQL mode into account. It treats strings delimited by ", ', and ` characters the same, regardless of whether ANSI_QUOTES is enabled.

? Within /*! ... */ and /*+ ... */ comments, the mysql client parser interprets short-form mysql commands. The server parser does not

interpret them because these commands have no meaning on the server side.

If it is desirable for mysql not to interpret short-form commands within comments, a partial workaround is to use the --binary-mode option, which causes all mysql commands to be disabled except \C and \d in noninteractive mode (for input piped to mysql or loaded using the source command).

COPYRIGHT

NOTES

1. MySQL Shell 8.0

   https://dev.mysql.com/doc/mysql-shell/8.0/en/

2. C API Basic Data Structures

   https://dev.mysql.com/doc/c-api/8.0/en/c-api-data-structures.html

3. Multiple Statement Execution Support

   https://dev.mysql.com/doc/c-api/8.0/en/c-api-multiple-queries.html

4. mysql_change_user()

   https://dev.mysql.com/doc/c-api/8.0/en/mysql-change-user.html

5. Automatic Reconnection Control

   https://dev.mysql.com/doc/c-api/8.0/en/c-api-auto-reconnect.html

SEE ALSO

For more information, please refer to the MySQL Reference Manual, which

may already be installed locally and which is also available online at

http://dev.mysql.com/doc/.

AUTHOR

Oracle Corporation (http://dev.mysql.com/).