8.6 Comment Syntax

MySQL Server supports three comment styles:

- From a “`#`” character to the end of the line.
- From a “`-- `” sequence to the end of the line. This style is supported as of MySQL 3.23.3. In MySQL, the “`-- `” (double-dash) comment style requires the second dash to be followed by at least one whitespace or control character (such as a space, tab, newline, and so on). This syntax differs slightly from standard SQL comment syntax, as discussed in Section 1.9.5.8, “'`--`' as the Start of a Comment”.
- From a `/*` sequence to the following `*/` sequence, as in the C programming language. This syntax enables a comment to extend over multiple lines because the beginning and closing sequences need not be on the same line.

The following example demonstrates all three comment styles:

```
mysql> SELECT 1+1;     # This comment continues to the end of line
mysql> SELECT 1+1;     -- This comment continues to the end of line
mysql> SELECT 1 /* this is an in-line comment */ + 1;
mysql> SELECT 1+
/*
this is a
multiple-line comment
*/
1;
```

Nested comments are not supported.

MySQL Server supports some variants of C-style comments. These enable you to write code that includes MySQL extensions, but is still portable, by using comments of the following form:

```
/*! MySQL-specific code */
```

In this case, MySQL Server parses and executes the code within the comment as it would any other SQL statement, but other SQL servers will ignore the extensions. For example, MySQL Server recognizes the `STRAIGHT_JOIN` keyword in the following statement, but other servers will not:

```
SELECT /*! STRAIGHT_JOIN */ col1 FROM table1,table2 WHERE ...
```

If you add a version number after the “`!`” character, the syntax within the comment is executed only if the MySQL version is greater than or equal to the specified version number. The `TEMPORARY` keyword in the following comment is executed only by servers from MySQL 3.23.02 or higher:

```
CREATE /*!32302 TEMPORARY */ TABLE t (a INT);
```

The comment syntax just described applies to how the **mysqld** server parses SQL statements. The **mysql** client program also performs some parsing of statements before sending them to the server. (It does this to determine statement boundaries within a multiple-statement input line.) However, there are some limitations on the way that **mysql** parses `/* ... */` comments:

- A semicolon within the comment is taken to indicate the end of the current SQL statement and anything following it to indicate the beginning of the next statement. This problem was fixed in MySQL 4.0.13.
- A single quote, double quote, or backtick character is taken to indicate the beginning of a quoted string or identifier, even within a comment. If the quote is not matched by a second quote within the comment, the parser doesn't realize the comment has ended. If you are running **mysql** interactively, you can tell that it has gotten confused like this because the prompt changes from `mysql>` to `'>`, `">`, or `` `> ``. This problem was fixed in MySQL 4.1.1.
- The use of short-form commands such as `\C` within multi-line `/* ... */` comments is not supported.

Comments in this format, `/*!12345 ... */`, are not stored on the server. If this format is used to comment stored routines, the comments will not be retained on the server.

For affected versions of MySQL, these limitations apply both when you run **mysql** interactively and when you put commands in a file and use **mysql** in batch mode to process the file with **mysql < *file_name***.

Previous / Next / Up / Table of Contents

User Comments

Posted by John Bercik on April 29 2009 6:27pm [Delete] [Edit]

Just a comment on the comments; It does not appear that you can add comments to Views in MySQL.

Posted by Andre K. on May 25 2009 1:56pm [Delete] [Edit]

It took me about half an hour to figure out that the semicolon problem isn't fixed in 4.0.13 (as stated in the manual) and at least exists up to 4.1.13.

This throws an "Error 1064":
/*50000 DROP FUNCTION IF EXISTS `foo`; */

But this works as expected (please note the positioning of the semicolon outside the comment):
/*50000 DROP FUNCTION IF EXISTS `foo` */;

(Tested with 4.1.13-max-log and mysql in batch mode)

Posted by Balaji Devarajan on December 9 2009 8:43pm [Delete] [Edit]

We have sql queries in 1000 places in our bash scripts and it was becoming challenging for me to figure out in the processlist to know which script is running the query. So wanted to come with some simple logic to show the script name as comments for each query in the processlist,

we have the connection string in our config file...it would be easy to change in one place instead changing at all the places in our script.

our project.config
```
MYSQL1="mysql -h $host -u$dbUser -p$dbPass -A $db"
function visible() {
read sql
case "$1" in
'MYSQL1')
MYSQL=$MYSQL1
;;
'MYSQL2')
MYSQL=$MYSQL2
;;
*)
exit 1
;;
esac

echo "$sql" | sed 's/^/\/* '`basename $0`' *\/ /g' | $MYSQL -c
}

MYSQL='visible MYSQL1'
```

the script(query_visible_test.sh) will still use what they are using.(something like this)
```
echo "select DISTINCT ID,count(1) CNT from STATS where HIT_DATE between '2009-11-01' and '2009-12-09' group by ID order by CNT desc limit 100;" | $MYSQL
```

now you will the script name in the processlist

```
| 987932 | user      | host:55111 | m6_agg  | Query   |       5 | Sorting for group                | /* query_visible_test.sh */ select DISTINCT ID,count(1) CNT from STATS where HIT_DATE between
```

Now I can go check the script or kill process if need...when the query gets locked or locking replication...etc.

Add your own comment

Top / Previous / Next / Up / Table of Contents

ORACLE © 2014, Oracle Corporation and/or its affiliates