NAME

mysqlbinlog - utility for processing binary log files

SYNOPSIS

mysqlbinlog [**options**] *log_file* **...**

DESCRIPTION

The servers binary log consists of files containing "events" that describe modifications to database contents. The server writes these files in binary format. To display their contents in text format, use the **mysqlbinlog** utility. You can also use **mysqlbinlog** to display the contents of relay log files written by a slave server in a replication setup because relay logs have the same format as binary logs. The binary log and relay log are discussed further in Section 5.4.4, "The Binary Log", and Section 17.2.2, "Replication Relay and Status Logs".

Invoke **mysqlbinlog** like this:

shell> **mysqlbinlog** [*options*] *log_file* **...**

For example, to display the contents of the binary log file named binlog.000003, use this command:

shell> **mysqlbinlog binlog.0000003**

The output includes events contained in binlog.000003. For statement-based logging, event information includes the SQL statement, the ID of the server on which it was executed, the timestamp when the statement was executed, how much time it took, and so forth. For row-based logging, the event indicates a row change rather than an SQL statement. See Section 17.1.2, "Replication Formats", for information about logging modes.

Events are preceded by header comments that provide additional information. For example:

```
# at 141
#100309 9:28:36 server id 123 end_log_pos 245
Query thread_id=3350 exec_time=11 error_code=0
```

In the first line, the number following at indicates the file offset, or starting position, of the event in the binary log file.

The second line starts with a date and time indicating when the statement started on the server where the event originated. For replication, this timestamp is propagated to slave servers. server id is the server_id value of the server where the event originated. end_log_pos indicates where the next event starts (that is, it is the end position of the current event + 1). thread_id indicates which thread executed the event. exec_time is the time spent executing the event, on a master server. On a slave, it is the difference of the end execution time on the slave minus the beginning execution time on the master. The difference serves as an indicator of how much replication lags behind the master. error_code indicates the result from executing the event. Zero means that no error occurred.

> **Note**
>
> When using event groups, the file offsets of events may be grouped together and the comments of events may be grouped together. Do not mistake these grouped events for blank file offsets.

The output from **mysqlbinlog** can be re-executed (for example, by using it as input to **mysql**) to redo the statements in the log. This is useful for recovery operations after a server crash. For other usage examples, see the discussion later in this section and in Section 7.5, "Point-in-Time (Incremental) Recovery Using the Binary Log".

Normally, you use **mysqlbinlog** to read binary log files directly and apply them to the local MySQL server. It is also possible to read binary logs from a remote server by using the **--read-from-remote-server** option. To read remote binary logs, the connection parameter options can be given to indicate how to connect to the server. These options are **--host**, **--password**, **--port**,

--protocol, **--socket**, and **--user**; they are ignored except when you also use the **--read-from-remote-server** option.

When running **mysqlbinlog** against a large binary log, be careful that the filesystem has enough space for the resulting files. To configure the directory that **mysqlbinlog** uses for temporary files, use the TMPDIR environment variable.

mysqlbinlog supports the following options, which can be specified on the command line or in the [mysqlbinlog] and [client] groups of an option file. For information about option files used by MySQL programs, see Section 4.2.6, "Using Option Files".

- **--help**, **-?**

  Display a help message and exit.
- **--base64-output**[=*value*]

  This option determines when events should be displayed encoded as base-64 strings using BINLOG statements. The option has these permissible values (not case-sensitive):
  - AUTO (automatic) or UNSPEC (unspecified) displays BINLOG statements automatically when necessary (that is, for format description events and row events). If no **--base64-output** option is given, the effect is the same as **--base64-output=AUTO**.

    Note

    Automatic BINLOG display is the only safe behavior if you intend to use the output of **mysqlbinlog** to re-execute binary log file contents. The other option values are intended only for debugging or testing purposes because they may produce output that does not include all events in executable form.
  - ALWAYS displays BINLOG statements whenever possible. If the **--base64-output** option is given without a value, the effect is the same as **--base64-output=ALWAYS**.

    Note

    Changes to replication in MySQL 5.6 make output generated by this option unusable, so ALWAYS is deprecated in MySQL 5.5 and will be an invalid value in MySQL 5.6
  - NEVER causes BINLOG statements not to be displayed. **mysqlbinlog** exits with an error if a row event is found that must be displayed using BINLOG.
  - DECODE-ROWS specifies to **mysqlbinlog** that you intend for row events to be decoded and displayed as commented SQL statements by also specifying the **--verbose** option. Like NEVER, DECODE-ROWS suppresses display of BINLOG statements, but unlike NEVER, it does not exit with an error if a row event is found.

  For examples that show the effect of **--base64-output** and **--verbose** on row event output, see the section called "MYSQLBINLOG ROW EVENT DISPLAY".
- **--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 10.14, "Character Set Configuration".
- **--database=***db_name*, **-d** *db_name*

  This option causes **mysqlbinlog** to output entries from the binary log (local log only) that occur while *db_name* is been selected as the default database by USE.

  The **--database** option for **mysqlbinlog** is similar to the **--binlog-do-db** option for **mysqld**, but can be used to specify only one database. If **--database** is given multiple times, only the last instance is used.

  The effects of this option depend on whether the statement-based or row-based logging format is in use, in the same way that the effects of **--binlog-do-db** depend on whether

statement-based or row-based logging is in use.

Statement-based logging. The **--database** option works as follows:

- While *db_name* is the default database, statements are output whether they modify tables in *db_name* or a different database.
- Unless *db_name* is selected as the default database, statements are not output, even if they modify tables in *db_name*.
- There is an exception for CREATE DATABASE, ALTER DATABASE, and DROP DATABASE. The database being *created, altered, or dropped* is considered to be the default database when determining whether to output the statement.

Suppose that the binary log was created by executing these statements using statement-based-logging:

```
INSERT INTO test.t1 (i) VALUES(100);
INSERT INTO db2.t2 (j) VALUES(200);
USE test;
INSERT INTO test.t1 (i) VALUES(101);
INSERT INTO t1 (i) VALUES(102);
INSERT INTO db2.t2 (j) VALUES(201);
USE db2;
INSERT INTO test.t1 (i) VALUES(103);
INSERT INTO db2.t2 (j) VALUES(202);
INSERT INTO t2 (j) VALUES(203);
```

mysqlbinlog --database=testdoes not output the first t wo INSERT statements because there is no default database. It outputs the three INSERT statements following USE test, but not the three INSERT statements following USE db2.

mysqlbinlog --database=db2does not output the first t wo INSERT statements because there is no default database. It does not output the three INSERT statements following USE test, but does output the three INSERT statements following USE db2.

Row-based logging. **mysqlbinlog** outputs only entries that change tables belonging to *db_name*. The default database has no effect on this. Suppose that the binary log just described was created using row-based logging rather than statement-based logging. **mysqlbinlog --database=test**outputs only those en tries that modify t1 in the test database, regardless of whether USE was issued or what the default database is. If a server is running with binlog_format set to MIXED and you want it to be possible to use **mysqlbinlog** with the **--database** option, you must ensure that tables that are modified are in the database selected by USE. (In particular, no cross-database updates should be used.)

> **Note**
>
> Prior to MySQL NDB Cluster 7.2.2, this option did not work correctly with NDB Cluster tables unless, unless the binary log was generated using **--log-bin-use-v1-row-events=0**. (Bug #13067813)

- **--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/mysqlbinlog.trace.
- **--debug-check**

  Print some debugging information when the program exits.
- **--debug-info**

  Print debugging information and memory and CPU usage statistics when the program exits.
- **--default-auth**=*plugin*

  A hint about the client-side authentication plugin to use. See Section 6.3.6, “Pluggable Authentication”.

This option was added in MySQL 5.5.10.

- **--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. *file_name* is interpreted relative to the current directory if given as a relative path name.

- **--defaults-file=***file_name*

  Use only the given option file. If the file does not exist or is otherwise inaccessible, an error occurs. *file_name* is interpreted relative to the current directory if given as a relative path name.

- **--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, **mysqlbinlog** normally reads the [client] and [mysqlbinlog] groups. If the **--defaults-group-suffix=_other** option is given, **mysqlbinlog** also reads the [client_other] and [mysqlbinlog_other] groups.

- **--disable-log-bin**, **-D**

  Disable binary logging. This is useful for avoiding an endless loop if you use the **--to-last-log** option and are sending the output to the same MySQL server. This option also is useful when restoring after a crash to avoid duplication of the statements you have logged.

  This option causes **mysqlbinlog** to include a SET sql_log_bin = 0 statement in its output to disable binary logging of the remaining output. Manipulating the session value of the sql_log_bin system variable is a restricted operation, so this option requires that you have privileges sufficient to set restricted session variables. See Section 5.1.8.1, "System Variable Privileges".

- **--force-if-open**, **-F**

  Read binary log files even if they are open or were not closed properly.

- **--force-read**, **-f**

  With this option, if **mysqlbinlog** reads a binary log event that it does not recognize, it prints a warning, ignores the event, and continues. Without this option, **mysqlbinlog** stops if it reads such an event.

- **--hexdump**, **-H**

  Display a hex dump of the log in comments, as described in the section called "MYSQLBINLOG HEX DUMP FORMAT". The hex output can be helpful for replication debugging.

- **--host=***host_name*, **-h** *host_name*

  Get the binary log from the MySQL server on the given host.

- **--local-load=***dir_name*, **-l** *dir_name*

  Prepare local temporary files for LOAD DATA INFILE in the specified directory.

  Important

  These temporary files are not automatically removed by **mysqlbinlog** or any other MySQL program.

- **--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.

- **--offset=***N*, **-o** *N*

  Skip the first *N* entries in the log.

- **--password**[=*password*], **-p**[*password*]

  The password to use when connecting to the server. If you use the short option form (**-p**), you *cannot* have a space between the option and the password. If you omit the *password* value following the **--password** or **-p** option on the command line, **mysqlbinlog** prompts

for one.

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

- **--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 **mysqlbinlog** does not find it. See Section 6.3.6, “Pluggable Authentication”.

  This option was added in MySQL 5.5.10.
- **--port=***port_num*, **-P** *port_num*

  The TCP/IP port number to use for connecting to a remote server.
- **--position=***N*

  Deprecated. Use **--start-position** instead. **--position** was removed in MySQL 5.5.3.
- **--print-defaults**

  Print the program name and all options that it gets from option files.
- **--protocol={TCP|SOCKET|PIPE|MEMORY}**

  The connection protocol to use for connecting to the server. It is useful when the other connection parameters normally would cause a protocol to be used other than the one you want. For details on the permissible values, see Section 4.2.2, “Connecting to the MySQL Server”.
- **--read-from-remote-server**, **-R**

  Read the binary log from a MySQL server rather than reading a local log file. Any connection parameter options are ignored unless this option is given as well. These options are **--host**, **--password**, **--port**, **--protocol**, **--socket**, and **--user**.

  This option requires that the remote server be running. It works only for binary log files on the remote server, not relay log files.
- **--result-file=***name*, **-r** *name*

  Direct output to the given file.
- **--server-id=***id*

  Display only those events created by the server having the given server ID.
- **--server-id-bits=***N*

  Use only the first *N* bits of the server_id to identify the server. If the binary log was written by a **mysqld** with server-id-bits set to less than 32 and user data stored in the most significant bit, running **mysqlbinlog** with **--server-id-bits** set to 32 enables this data to be seen.

  This option is supported only by the versions of **mysqlbinlog** supplied with the NDB Cluster distribution, or built from the NDB Cluster sources.
- **--set-charset=***charset_name*

  Add a SET NAMES *charset_name* statement to the output to specify the character set to be used for processing log files.
- **--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.

  The server must be started with the **--shared-memory** option to enable shared-memory connections.
- **--short-form**, **-s**

  Display only the statements contained in the log, without any extra information or row-

based events. This is for testing only, and should not be used in production systems.

- **--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.

- **--ssl***

  Options that begin with **--ssl** specify whether to connect to the server using SSL and indicate where to find SSL keys and certificates. See Section 6.4.2, "Command Options for Encrypted Connections".

- **--start-datetime**=*datetime*

  Start reading the binary log at the first event having a timestamp equal to or later than the *datetime* argument. The *datetime* value is relative to the local time zone on the machine where you run **mysqlbinlog**. The value should be in a format accepted for the DATETIME or TIMESTAMP data types. For example:

  shell> **mysqlbinlog --start-datetime="2005-12-25 11:25:56" binlog.000003**

  This option is useful for point-in-time recovery. See Section 7.3, "Example Backup and Recovery Strategy".

- **--start-position**=*N*, **-j** *N*

  Start reading the binary log at the first event having a position equal to or greater than *N*. This option applies to the first log file named on the command line.

  This option is useful for point-in-time recovery. See Section 7.3, "Example Backup and Recovery Strategy".

- **--stop-datetime**=*datetime*

  Stop reading the binary log at the first event having a timestamp equal to or later than the *datetime* argument. This option is useful for point-in-time recovery. See the description of the **--start-datetime** option for information about the *datetime* value.

  This option is useful for point-in-time recovery. See Section 7.3, "Example Backup and Recovery Strategy".

- **--stop-position**=*N*

  Stop reading the binary log at the first event having a position equal to or greater than *N*. This option applies to the last log file named on the command line.

  This option is useful for point-in-time recovery. See Section 7.3, "Example Backup and Recovery Strategy".

- **--to-last-log**, **-t**

  Do not stop at the end of the requested binary log from a MySQL server, but rather continue printing until the end of the last binary log. If you send the output to the same MySQL server, this may lead to an endless loop. This option requires **--read-from-remote-server**.

- **--user**=*user_name*, **-u** *user_name*

  The MySQL user name to use when connecting to a remote server.

- **--verbose**, **-v**

  Reconstruct row events and display them as commented SQL statements. If this option is given twice, the output includes comments to indicate column data types and some metadata.

  For examples that show the effect of **--base64-output** and **--verbose** on row event output, see the section called "MYSQLBINLOG ROW EVENT DISPLAY".

- **--version**, **-V**

  Display version information and exit.

In MySQL 5.5, the version number shown for **mysqlbinlog** is always 3.3.

You can also set the following variable by using **--***var_name*=*value* syntax:

- open_files_limit

  Specify the number of open file descriptors to reserve.

You can pipe the output of **mysqlbinlog** into the **mysql** client to execute the events contained in the binary log. This technique is used to recover from a crash when you have an old backup (see Section 7.5, "Point-in-Time (Incremental) Recovery Using the Binary Log"). For example:

shell> **mysqlbinlog binlog.000001 | mysql -u root -p**

Or:

shell> **mysqlbinlog binlog.[0-9]* | mysql -u root -p**

You can also redirect the output of **mysqlbinlog** to a text file instead, if you need to modify the statement log first (for example, to remove statements that you do not want to execute for some reason). After editing the file, execute the statements that it contains by using it as input to the **mysql** program:

shell> **mysqlbinlog binlog.000001 > tmpfile**
shell> *... edit tmpfile...*
shell> **mysql -u root -p < tmpfile**

When **mysqlbinlog** is invoked with the **--start-position** option, it displays only those events with an offset in the binary log greater than or equal to a given position (the given position must match the start of one event). It also has options to stop and start when it sees an event with a given date and time. This enables you to perform point-in-time recovery using the **--stop-datetime** option (to be able to say, for example, "roll forward my databases to how they were today at 10:30 a.m.").

If you have more than one binary log to execute on the MySQL server, the safe method is to process them all using a single connection to the server. Here is an example that demonstrates what may be *unsafe*:

shell> **mysqlbinlog binlog.000001 | mysql -u root -p # DANGER!!**
shell> **mysqlbinlog binlog.000002 | mysql -u root -p # DANGER!!**

Processing binary logs this way using multiple connections to the server causes problems if the first log file contains a CREATE TEMPORARY TABLE statement and the second log contains a statement that uses the temporary table. When the first **mysql** process terminates, the server drops the temporary table. When the second **mysql** process attempts to use the table, the server reports "unknown table."

To avoid problems like this, use a *single* **mysql** process to execute the contents of all binary logs that you want to process. Here is one way to do so:

shell> **mysqlbinlog binlog.000001 binlog.000002 | mysql -u root -p**

Another approach is to write all the logs to a single file and then process the file:

shell> **mysqlbinlog binlog.000001 > /tmp/statements.sql**
shell> **mysqlbinlog binlog.000002 >> /tmp/statements.sql**
shell> **mysql -u root -p -e "source /tmp/statements.sql"**

mysqlbinlog can produce output that reproduces a LOAD DATA INFILE operation without the original data file. **mysqlbinlog** copies the data to a temporary file and writes a LOAD DATA LOCAL INFILE statement that refers to the file. The default location of the directory where these files are written is system-specific. To specify a directory explicitly, use the **--local-load** option.

Because **mysqlbinlog** converts LOAD DATA INFILE statements to LOAD DATA LOCAL INFILE statements (that is, it adds LOCAL), both the client and the server that you use to

process the statements must be configured with the LOCAL capability enabled. See Section 6.1.6, "Security Issues with LOAD DATA LOCAL".

Warning

The temporary files created for LOAD DATA LOCAL statements are *not* automatically deleted because they are needed until you actually execute those statements. You should delete the temporary files yourself after you no longer need the statement log. The files can be found in the temporary file directory and have names like *original_file_name-#-#*.

MYSQLBINLOG HEX DUMP FORMAT

The **--hexdump** option causes **mysqlbinlog** to produce a hex dump of the binary log contents:

shell> **mysqlbinlog --hexdump master-bin.000001**

The hex output consists of comment lines beginning with #, so the output might look like this for the preceding command:

```
/*!40019 SET @@session.max_insert_delayed_threads=0*/;
/*!50003 SET @OLD_COMPLETION_TYPE=@@COMPLETION_TYPE,COMPLETION_TYPE=0*/;
# at 4
#051024 17:24:13 server id 1 end_log_pos 98
# Position Timestamp Type Master ID Size Master Pos Flags
# 00000004 9d fc 5c 43 0f 01 00 00 00 5e 00 00 00 62 00 00 00 00 00
# 00000017 04 00 35 2e 30 2e 31 35 2d 64 65 62 75 67 2d 6c |..5.0.15.debug.l|
# 00000027 6f 67 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |og..............|
# 00000037 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
# 00000047 00 00 00 00 9d fc 5c 43 13 38 0d 00 08 00 12 00 |.......C.8......|
# 00000057 04 04 04 04 12 00 00 4b 00 04 1a |.......K...|
# Start: binlog v 4, server v 5.0.15-debug-log created 051024 17:24:13
# at startup
ROLLBACK;
```

Hex dump output currently contains the elements in the following list. This format is subject to change. For more information about binary log format, see **MySQL Internals: The Binary Log**[1].

- Position: The byte position within the log file.
- Timestamp: The event timestamp. In the example shown, 9d fc 5c 43 is the representation of 051024 17:24:13 in hexadecimal.
- Type: The event type code.
- Master ID: The server ID of the master that created the event.
- Size: The size in bytes of the event.
- Master Pos: The position of the next event in the original master log file.
- Flags: Event flag values.

MYSQLBINLOG ROW EVENT DISPLAY

The following examples illustrate how **mysqlbinlog** displays row events that specify data modifications. These correspond to events with the WRITE_ROWS_EVENT, UPDATE_ROWS_EVENT, and DELETE_ROWS_EVENT type codes. The **--base64-output=DECODE-ROWS** and **--verbose** options may be used to affect row event output.

Suppose that the server is using row-based binary logging and that you execute the following sequence of statements:

```
CREATE TABLE t
(
id INT NOT NULL,
name VARCHAR(20) NOT NULL,
date DATE NULL
) ENGINE = InnoDB;
```

```
START TRANSACTION;
INSERT INTO t VALUES(1, apple, NULL);
UPDATE t SET name = pear, date = 2009-01-01 WHERE id = 1;
DELETE FROM t WHERE id = 1;
COMMIT;
```

By default, **mysqlbinlog** displays row events encoded as base-64 strings using BINLOG statements. Omitting extraneous lines, the output for the row events produced by the preceding statement sequence looks like this:

```
shell> mysqlbinlog log_file
...
# at 218
#080828 15:03:08 server id 1 end_log_pos 258 Write_rows: table id 17 flags: STMT_END_F
BINLOG
fAS3SBMBAAAALAAAANoAAAAAABEAAAAAAAAAABHRlc3QAAXQAAwMPCgIUAAQ=
fAS3SBcBAAAAKAAAAAIBAAAQABEAAAAAAAEAA//8AQAAAAVhcHBsZQ==
/*!*/;
...
# at 302
#080828 15:03:08 server id 1 end_log_pos 356 Update_rows: table id 17 flags: STMT_END_F
BINLOG
fAS3SBMBAAAALAAAAC4BAAAAABEAAAAAAAAAABHRlc3QAAXQAAwMPCgIUAAQ=
fAS3SBgBAAAANgAAAGQBAAAQABEAAAAAAAEAA////AEAAAAFYXBwbGX4AQAAAARwZWFyIbI
/*!*/;
...
# at 400
#080828 15:03:08 server id 1 end_log_pos 442 Delete_rows: table id 17 flags: STMT_END_F
BINLOG
fAS3SBMBAAAALAAAAJABAAAAABEAAAAAAAAAABHRlc3QAAXQAAwMPCgIUAAQ=
fAS3SBkBAAAAKgAAALoBAAAQABEAAAAAAAEAA//4AQAAAARwZWFyIbIP
/*!*/;
```

To see the row events as comments in the form of "pseudo-SQL" statements, run **mysqlbinlog** with the **--verbose** or **-v** option. The output will contain lines beginning with ###:

```
shell> mysqlbinlog -v log_file
...
# at 218
#080828 15:03:08 server id 1 end_log_pos 258 Write_rows: table id 17 flags: STMT_END_F
BINLOG
fAS3SBMBAAAALAAAANoAAAAAABEAAAAAAAAAABHRlc3QAAXQAAwMPCgIUAAQ=
fAS3SBcBAAAAKAAAAAIBAAAQABEAAAAAAAEAA//8AQAAAAVhcHBsZQ==
/*!*/;
### INSERT INTO test.t
### SET
### @1=1
### @2=apple
### @3=NULL
...
# at 302
#080828 15:03:08 server id 1 end_log_pos 356 Update_rows: table id 17 flags: STMT_END_F
BINLOG
fAS3SBMBAAAALAAAAC4BAAAAABEAAAAAAAAAABHRlc3QAAXQAAwMPCgIUAAQ=
fAS3SBgBAAAANgAAAGQBAAAQABEAAAAAAAEAA////AEAAAAFYXBwbGX4AQAAAARwZWFyIbI
/*!*/;
### UPDATE test.t
```

```
### WHERE
### @1=1
### @2=apple
### @3=NULL
### SET
### @1=1
### @2=pear
### @3=2009:01:01
...
# at 400
#080828 15:03:08 server id 1 end_log_pos 442 Delete_rows: table id 17 flags: STMT_END_F
BINLOG
fAS3SBMBAAAALAAAAJABAAAAABEAAAAAAAAABHRlc3QAAXQAAwMPCgIUAAQ=
fAS3SBkBAAAAKgAAALoBAAAQABEAAAAAAAEAA//4AQAAAARwZWFyIbIP
/*!*/;
### DELETE FROM test.t
### WHERE
### @1=1
### @2=pear
### @3=2009:01:01
```

Specify **--verbose** or **-v** twice to also display data types and some metadata for each column. The output will contain an additional comment following each column change:

shell> **mysqlbinlog -vv** *log_file*

```
...
# at 218
#080828 15:03:08 server id 1 end_log_pos 258 Write_rows: table id 17 flags: STMT_END_F
BINLOG
fAS3SBMBAAAALAAAANoAAAAAABEAAAAAAAAABHRlc3QAAXQAAwMPCgIUAAQ=
fAS3SBcBAAAAKAAAAAIBAAAQABEAAAAAAAEAA//8AQAAAAVhcHBsZQ==
/*!*/;
### INSERT INTO test.t
### SET
### @1=1 /* INT meta=0 nullable=0 is_null=0 */
### @2=apple /* VARSTRING(20) meta=20 nullable=0 is_null=0 */
### @3=NULL /* VARSTRING(20) meta=0 nullable=1 is_null=1 */
...
# at 302
#080828 15:03:08 server id 1 end_log_pos 356 Update_rows: table id 17 flags: STMT_END_F
BINLOG
fAS3SBMBAAAALAAAAC4BAAAAABEAAAAAAAAABHRlc3QAAXQAAwMPCgIUAAQ=
fAS3SBgBAAAANgAAAGQBAAAQABEAAAAAAAEAA////AEAAAAFYXBwbGX4AQAAAARwZWFyIbI
/*!*/;
### UPDATE test.t
### WHERE
### @1=1 /* INT meta=0 nullable=0 is_null=0 */
### @2=apple /* VARSTRING(20) meta=20 nullable=0 is_null=0 */
### @3=NULL /* VARSTRING(20) meta=0 nullable=1 is_null=1 */
### SET
### @1=1 /* INT meta=0 nullable=0 is_null=0 */
### @2=pear /* VARSTRING(20) meta=20 nullable=0 is_null=0 */
### @3=2009:01:01 /* DATE meta=0 nullable=1 is_null=0 */
...
# at 400
#080828 15:03:08 server id 1 end_log_pos 442 Delete_rows: table id 17 flags: STMT_END_F
```

```
BINLOG
fAS3SBMBAAAALAAAAJABAAAAABEAAAAAAAAABHRlc3QAAXQAAwMPCgIUAAQ=
fAS3SBkBAAAAKgAAALoBAAAQABEAAAAAAAEAA//4AQAAAARwZWFyIbIP
/*!*/;
### DELETE FROM test.t
### WHERE
### @1=1 /* INT meta=0 nullable=0 is_null=0 */
### @2=pear /* VARSTRING(20) meta=20 nullable=0 is_null=0 */
### @3=2009:01:01 /* DATE meta=0 nullable=1 is_null=0 */
```

You can tell **mysqlbinlog** to suppress the BINLOG statements for row events by using the **--base64-output=DECODE-ROWS** option. This is similar to **--base64-output=NEVER** but does not exit with an error if a row event is found. The combination of **--base64-output=DECODE-ROWS** and **--verbose** provides a convenient way to see row events only as SQL statements:

```
shell> mysqlbinlog -v --base64-output=DECODE-ROWS log_file
...
# at 218
#080828 15:03:08 server id 1 end_log_pos 258 Write_rows: table id 17 flags: STMT_END_F
### INSERT INTO test.t
### SET
### @1=1
### @2=apple
### @3=NULL
...
# at 302
#080828 15:03:08 server id 1 end_log_pos 356 Update_rows: table id 17 flags: STMT_END_F
### UPDATE test.t
### WHERE
### @1=1
### @2=apple
### @3=NULL
### SET
### @1=1
### @2=pear
### @3=2009:01:01
...
# at 400
#080828 15:03:08 server id 1 end_log_pos 442 Delete_rows: table id 17 flags: STMT_END_F
### DELETE FROM test.t
### WHERE
### @1=1
### @2=pear
### @3=2009:01:01
```

Note

You should not suppress BINLOG statements if you intend to re-execute **mysqlbinlog** output.

The SQL statements produced by **--verbose** for row events are much more readable than the corresponding BINLOG statements. However, they do not correspond exactly to the original SQL statements that generated the events. The following limitations apply:

- The original column names are lost and replaced by @*N*, where *N* is a column number.
- Character set information is not available in the binary log, which affects string column display:

- There is no distinction made between corresponding binary and nonbinary string types (BINARY and CHAR, VARBINARY and VARCHAR, BLOB and TEXT). The output uses a data type of STRING for fixed-length strings and VARSTRING for variable-length strings.
- For multibyte character sets, the maximum number of bytes per character is not present in the binary log, so the length for string types is displayed in bytes rather than in characters. For example, STRING(4) will be used as the data type for values from either of these column types:

  CHAR(4) CHARACTER SET latin1
  CHAR(2) CHARACTER SET ucs2
- Due to the storage format for events of type UPDATE_ROWS_EVENT, UPDATE statements are displayed with the WHERE clause preceding the SET clause.

Proper interpretation of row events requires the information from the format description event at the beginning of the binary log. Because **mysqlbinlog** does not know in advance whether the rest of the log contains row events, by default it displays the format description event using a BINLOG statement in the initial part of the output.

If the binary log is known not to contain any events requiring a BINLOG statement (that is, no row events), the **--base64-output=NEVER** option can be used to prevent this header from being written.

COPYRIGHT

NOTES

1. MySQL Internals: The Binary Log
   http://dev.mysql.com/doc/internals/en/binary-log.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/).