The events_statements_history table has the same columns and indexing as events_statements_current. See Section 27.12.6.1, “The events_statements_current Table”.

TRUNCATE TABLE is permitted for the events_statements_history table. It removes the rows.

For more information about the relationship between the three events_statements_*xxx* event tables, see Section 27.9, “Performance Schema Tables for Current and Historical Events”.

For information about configuring whether to collect statement events, see Section 27.12.6, “Performance Schema Statement Event Tables”.

27.12.6.3 The events_statements_history_long Table

The events_statements_history_long table contains the *N* most recent statement events that have ended globally, across all threads. Statement events are not added to the table until they have ended. When the table becomes full, the oldest row is discarded when a new row is added, regardless of which thread generated either row.

The value of *N* is autosized at server startup. To set the table size explicitly, set the performance_schema_events_statements_history_long_size system variable at server startup.

The events_statements_history_long table has the same columns as events_statements_current. See Section 27.12.6.1, “The events_statements_current Table”. Unlike events_statements_current, events_statements_history_long has no indexing.

TRUNCATE TABLE is permitted for the events_statements_history_long table. It removes the rows.

For more information about the relationship between the three events_statements_*xxx* event tables, see Section 27.9, “Performance Schema Tables for Current and Historical Events”.

For information about configuring whether to collect statement events, see Section 27.12.6, “Performance Schema Statement Event Tables”.

27.12.6.4 The prepared_statements_instances Table

The Performance Schema provides instrumentation for prepared statements, for which there are two protocols:

- The binary protocol. This is accessed through the MySQL C API and maps onto underlying server commands as shown in the following table.

C API Function	Corresponding Server Command
mysql_stmt_prepare()	COM_STMT_PREPARE
mysql_stmt_execute()	COM_STMT_EXECUTE
mysql_stmt_close()	COM_STMT_CLOSE

- The text protocol. This is accessed using SQL statements and maps onto underlying server commands as shown in the following table.

SQL Statement	Corresponding Server Command
PREPARE	SQLCOM_PREPARE
EXECUTE	SQLCOM_EXECUTE