

MySQL Performance Schema in Action

November, 5, 2018

Sveta Smirnova, Alexander Rubin with Nickolay Ihalainen



Table of Contents

- Internal Diagnostic in MySQL
- Configuration
- 5.6+: Statements Instrumentation
- 5.7+: Prepared Statements
- 5.7+: Stored Routines
- 5.7+: Locks Diagnostic
- 5.7+: Memory Usage
- 5.7+: Replication
- 5.7+: Variables
- 8.0+: Errors Summary

Who We Are



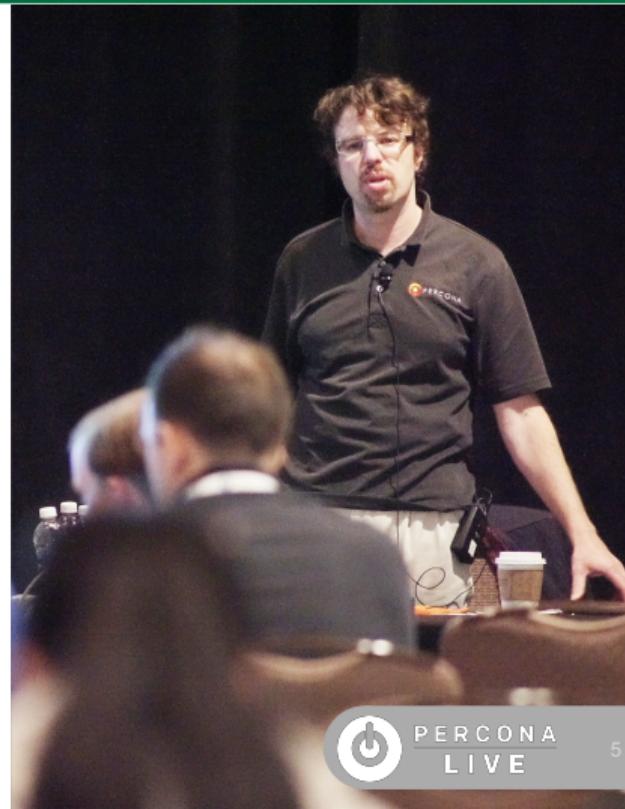
Who We Are: Svetlana Smirnova



- MySQL Support engineer
- Author of
 - MySQL Troubleshooting
 - JSON UDF functions
 - FILTER clause for MySQL
- Speaker
 - Percona Live, OOW, Fosdem, DevConf, HighLoad...

Who We Are: Alexander Rubin

- Principal Consultant in Architecture & Projects
- With MySQL for > 12 years
- Speaker
 - Percona Live, OOW, HighLoad...



Who We Are: Nickolay Ihalainen



- Senior Support Engineer
- In Percona for 7+ years
- DBA Expert in
 - MySQL
 - MongoDB
 - PostgreSQL

Internal Diagnostic in MySQL



Old Style

- INFORMATION_SCHEMA
- SHOW commands
 - Metadata
 - Storage engine extention
 - Runtime
 - Can be anything

Old Style

- INFORMATION_SCHEMA
- SHOW commands
- Status variables
 - Runtime

Old Style

- INFORMATION_SCHEMA
- SHOW commands
- Status variables
- Administration statements
 - ANALYZE
 - EXPLAIN
 - CHECK

Old Style

- INFORMATION_SCHEMA
- SHOW commands
- Status variables
- Administration statements
- No or limited information on
 - Performance
 - Internal server operations

Performance Schema

- Introduced in version 5.5

Performance Schema

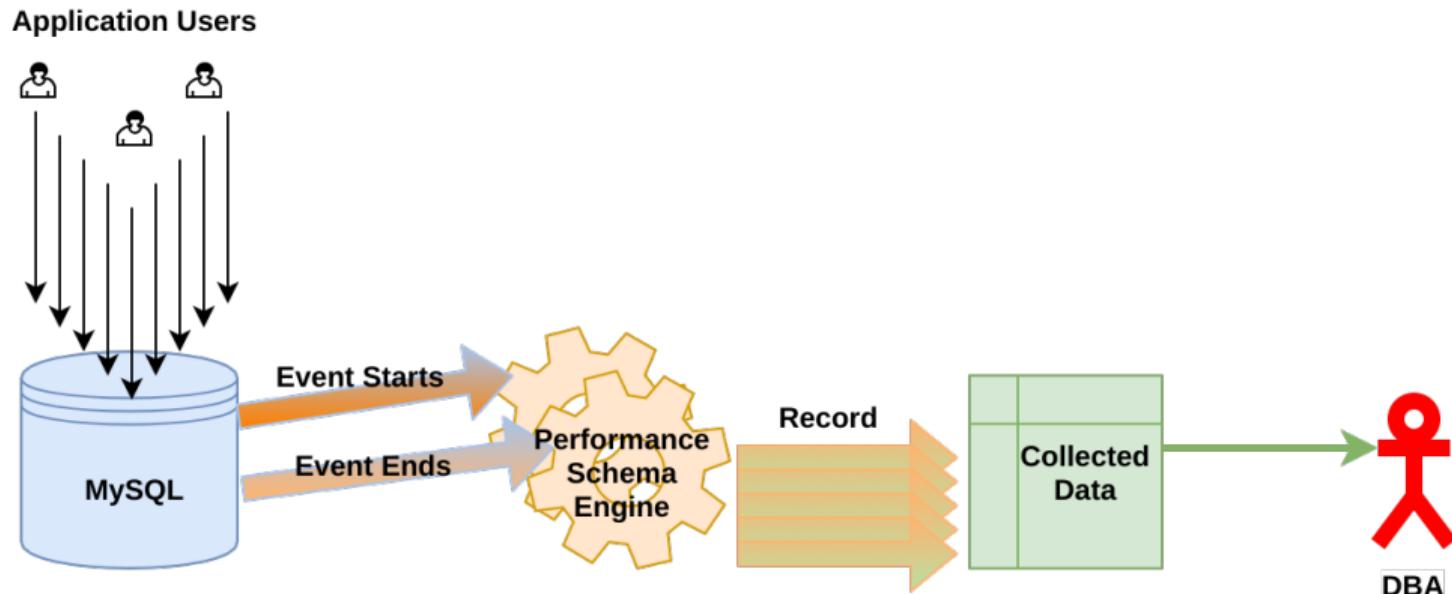
- Introduced in version 5.5
- Runtime performances statistics

Performance Schema

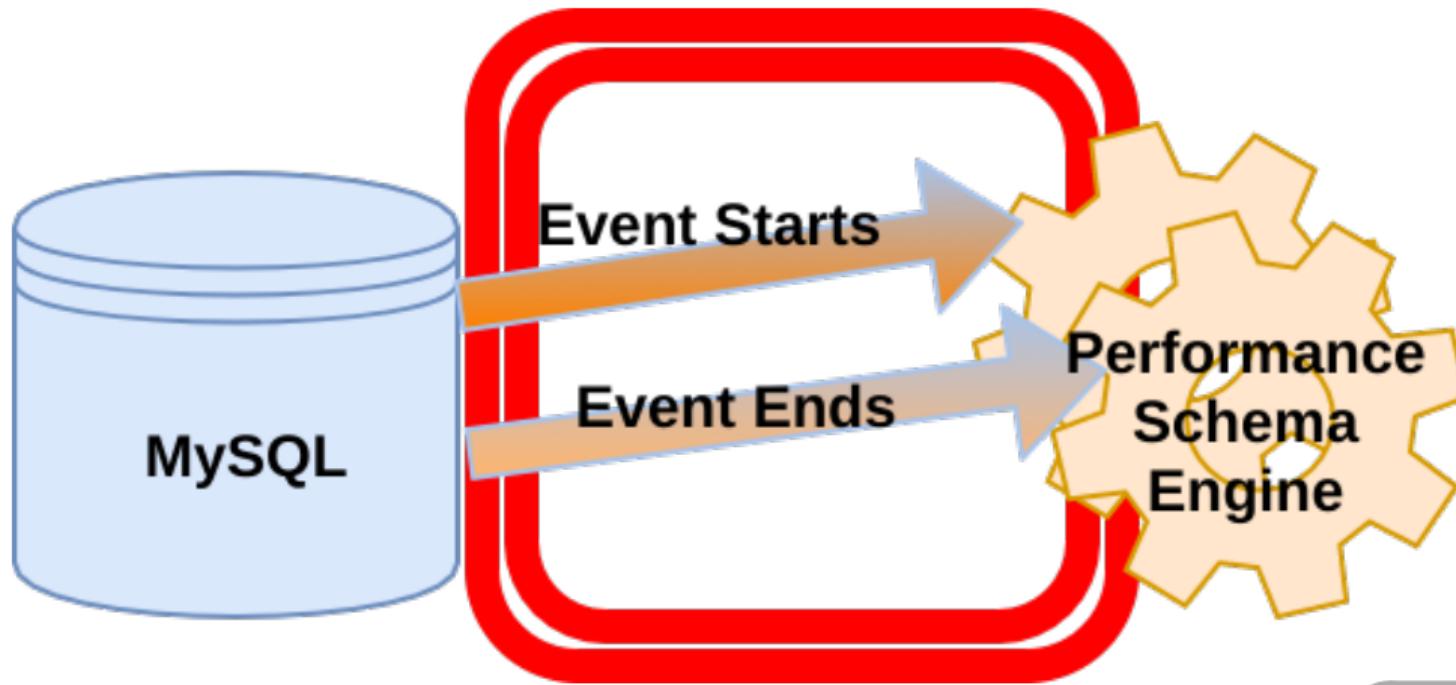
- Introduced in version 5.5
- Runtime performances statistics
- Wraps the diagnosed code

```
locker = PSI_RWLOCK_CALL(start_rwlock_wrwait)(  
    &state, lock->pfs_psi, PSI_RWLOCK_TRYEXCLUSIVELOCK,  
    file_name, static_cast<uint>(line));  
  
ret = rw_lock_x_lock_func_nowait(lock, file_name, line);  
  
if (locker != NULL)  
    PSI_RWLOCK_CALL(end_rwlock_wrwait)(  
        locker, static_cast<int>(ret));
```

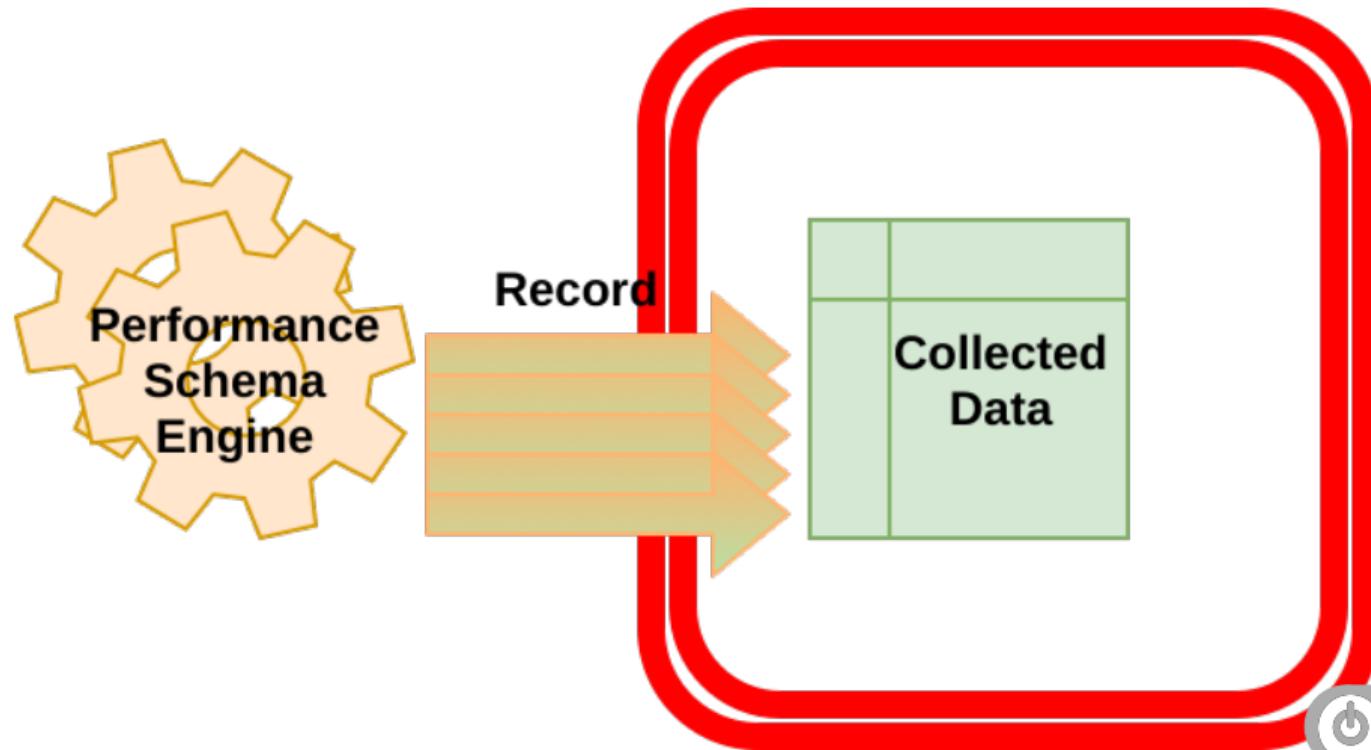
How Does Performance Schema Works?



Instruments: Get the Data



Consumers: Store the Data



Performance Schema Limitations

- Must be supported by the component
- Collects data only after they are enabled
- Never frees allocated memory

sys Schema

- Views on Performance Schema tables
- Stored routines

sys Schema

- Views on Performance Schema tables
- Stored routines
- Easier configuration
- Shortcuts to typical use cases

sys Schema

- Views on Performance Schema tables
- Stored routines
- Easier configuration
- Shortcuts to typical use cases
- 5.7+: Installed by default
- Before: github.com/mysql/mysql-sys

What will We Discuss Today?

- Statements
- Locks
- Memory Usage
- Replication
- Variables
- Errors

Configuration



What is Inside?

5.6

- 52 tables
- 561 instrs
- 589 Percona Server
- 31 variables

5.7

- 87 tables
- 1029 instrs
- 1067 Percona Server
- 42 variables

8.0

- 102 tables
- 1202 instrs
- 1229 Percona Server
- 44 variables

Performance Schema Defaults

- Defaults
 - ON

5.7: Only global, thread, statements and transactions instrumentation

8.0: Memory and MDL

Performance Schema Defaults

- Defaults

- ON

5.7: Only global, thread, statements and transactions instrumentation

8.0: Memory and MDL

- All other instruments/consumers disabled

Prepare

- We will turn required instrumentation ON for each exercise separately

Prepare

- We will turn required instrumentation ON for each exercise separately
- We will use pattern

```
update performance_schema.setup_consumers set enabled='yes'  
where name like 'OUR_REQUIREMENT_%';
```

```
update performance_schema.setup_instruments set enabled='yes', timed='yes'  
where name like 'OUR_REQUIREMENT_%';
```

Prepare

- We will turn required instrumentation ON for each exercise separately
- Or easier

```
call sys.ps_setup_enable_consumer(YOUR_CONSUMER);
```

```
call sys.ps_setup_enable_instrument(YOUR_INSTRUMENT);
```

Prepare

- We will turn required instrumentation ON for each exercise separately
- Be careful!
 - They are memory and CPU intensive
 - Do not turn them all ON until needed

5.6+: Statements Instrumentation



Statements: configuration

- Instruments

```
mysql> select name from setup_instruments where name like 'statement%';
+-----+
| name
+-----+
| statement/sql/select
| statement/sql/create_table
| statement/sql/create_index
| statement/sql/alter_table
| statement/sql/update
| statement/sql/insert
| statement/sql/insert_select
...
...
```

- Enable those you want to examine

Statements: configuration

- Instruments
- Consumers

```
mysql> select name from setup_consumers where name like '%statement%';
+-----+
| name          |
+-----+
| events_statements_current | -- Current statements
| events_statements_history | -- Last 10 statements
| events_statements_history_long | -- Last 1K statements
| statements_digest          | -- Digests
+-----+
4 rows in set (0.00 sec)
```

What Can We Discover?

- Why statements are slow?
 - Examine more rows than return/change
 - Use disk instead of memory
 - Full table scan instead of index
 - This is not full list!

What Can We Discover?

- Why statements are slow?
- Performance Schema has
 - Per-query statistics
 - Most evolving stages

Why Statements are Slow?

- `events_statements_*` and `prepared_statements_instances` tables
 - Important field names
 - `CREATED_TMP_DISK_TABLES`
 - `CREATED_TMP_TABLES`
 - `SELECT_FULL_JOIN`
 - `SELECT_RANGE_CHECK`
 - `SELECT_SCAN`
 - `SORT_MERGE_PASSES`
 - `SORT_SCAN`

Why Statements are Slow?

- `events_statements_*` and `prepared_statements_instances` tables
- Views in sys schema
 - Important view names
 - `statement_analysis`
 - `statements_with_full_table_scans`
 - `statements_with_runtimes_in_95th_percentile`
 - `statements_with_sorting`
 - `statements_with_temp_tables`
 - `statements_with_errors_or_warnings`

Why Statements are Slow?

- `events_statements_*` and `prepared_statements_instances` tables
- Views in sys schema
- Digest tables
 - Combined statistics
 - `events_statements_summary_by_account_by_event_name`
 - `events_statements_summary_by_host_by_event_name`
 - `events_statements_summary_by_thread_by_event_name`
 - `events_statements_summary_by_user_by_event_name`
 - `events_statements_summary_global_by_event_name`
 - **5.7+:** `events_statements_summary_by_program`

Why Statements are Slow?

- `events_statements_*` and `prepared_statements_instances` tables
- Views in sys schema
- Digest tables
 - Histogram Summary
 - `events_statements_histogram_by_digest`
 - `events_statements_histogram_global`
 - Do NOT mix with Optimizer histograms!

Why Statements are Slow?

- events_statements_* and prepared_statements_instances tables
- Views in sys schema
- Digest tables
 - events_statements_summary_by_digest
 - SCHEMA_NAME
 - DIGEST
 - DIGEST_TEXT
 - 8.0+: QUERY_SAMPLE_TEXT

Which Queries Do Not Use Indexes?

```
mysql> SELECT THREAD_ID TID, SUBSTR(SQL_TEXT, 1, 50) SQL_TEXT, ROWS_SENT RS,
-> ROWS_EXAMINED RE,CREATED_TMP_TABLES,NO_INDEX_USED,NO_GOOD_INDEX_USED
-> FROM performance_schema.events_statements_history
-> WHERE NO_INDEX_USED=1 OR NO_GOOD_INDEX_USED=1\G
***** 1. row *****
      TID: 10124
      SQL_TEXT: select emp_no, first_name, last_name from employee
      RS: 97750
      RE: 397774
CREATED_TMP_TABLES: 0
      NO_INDEX_USED: 1
NO_GOOD_INDEX_USED: 0
...
```

Take it Easy: Index Usage with sys Schema

```
mysql> SELECT query, total_latency, no_index_used_count, rows_sent,
-> rows_examined
-> FROM sys.statements_with_full_table_scans
-> WHERE db='employees' AND query NOT LIKE '%performance_schema%\G
***** 1. row *****
      query: SELECT COUNT ( `emp_no` ) FROM ... `emp_no`
              WHERE `title` = ?
      total_latency: 805.37 ms
no_index_used_count: 1
      rows_sent: 1
      rows_examined: 397774
...
```

Take it Easy: with Digest Tables

```
mysql> select DIGEST, DIGEST_TEXT, COUNT_STAR, SUM_CREATED_TMP_DISK_TABLES,
-> SUM_SELECT_FULL_JOIN, SUM_SELECT_RANGE , SUM_SELECT_SCAN , SUM_NO_INDEX_USED
-> SUM_ROWS_SENT, SUM_ROWS_EXAMINED
-> from events_statements_summary_by_digest where SUM_NO_INDEX_USED > 0\G
***** 1. row *****
          DIGEST: 3884185b07312b354c4918f2368d8fe2c431aeb8e39bf8ff5c3dcct
          DIGEST_TEXT: SELECT 'c' FROM 'sbtest1' WHERE 'id' BETWEEN ? AND ?
          COUNT_STAR: 1501791
SUM_CREATED_TMP_DISK_TABLES: 0
          SUM_SELECT_FULL_JOIN: 0
          SUM_SELECT_RANGE: 1501840
          SUM_SELECT_SCAN: 4
SUM_NO_INDEX_USED: 4
          SUM_ROWS_SENT: 150872400
SUM_ROWS_EXAMINED: 152872000
...
...
```

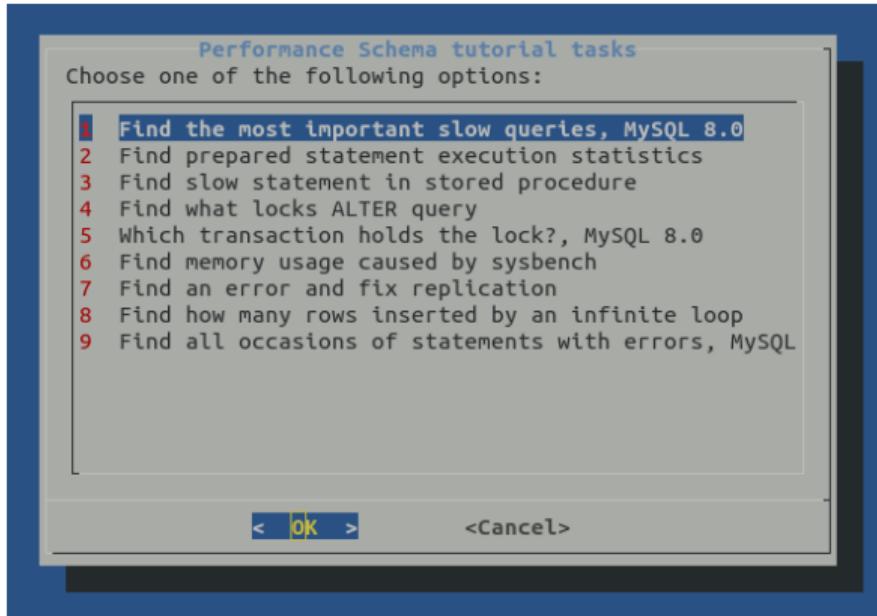


Statements: practice

- Login into EC2 instance
 - Login: see your card
 - Password: see your card

Statements: practice

- Start the task



Statements: practice

- Inside MySQL client

```
CALL help_task()\G
```

```
CALL help_solve()\G
```

- We need to find slowest queries:
 - With largest response time
 - With large scanned rows number
 - Not using indexes
 - Creating temporary tables

Statements Deep Dive

- events_stages_* tables

Statements Deep Dive

- events_stages_* tables
- Same information as in table INFORMATION_SCHEMA.PROCESSLIST or SHOW PROCESSLIST output
 - init
 - executing
 - Opening tables

Statements Deep Dive

- events_stages_* tables
- Same information as in table INFORMATION_SCHEMA.PROCESSLIST or SHOW PROCESSLIST output
 - init
 - executing
 - Opening tables
- Replacement for SHOW PROFILE

Statements Deep Dive

- events_stages_* tables
- Same information as in table INFORMATION_SCHEMA.PROCESSLIST or SHOW PROCESSLIST output
 - init
 - executing
 - Opening tables
- Replacement for SHOW PROFILE
- Only server-level
 - No storage engine information!

Stages: configuration

- Instruments

```
mysql> select name from setup_instruments where name like 'stage%';
+-----+
| name
+-----+
| stage/sql/After create
| stage/sql/allocating local table
| stage/sql/preparing for alter table
| stage/sql/altering table
| stage/sql/committing alter table to storage engine
| stage/sql/Changing master
| stage/sql/Checking master version
...
...
```

- Enable those you want to examine

Stages: configuration

- Instruments
- Consumers

```
mysql> select name from setup_consumers where name like '%stage%';
+-----+
| name           |
+-----+
| events_stages_current | -- Current stages
| events_stages_history | -- Last 10 stages
| events_stages_history_long | -- Last 1K stages
+-----+
3 rows in set (0.00 sec)
```

Stages Shortcuts

- Everything, related to temporary tables
 - EVENT_NAME LIKE 'stage/sql/%tmp%'
- Everything, related to locks
 - EVENT_NAME LIKE 'stage/sql/%lock%'
- Everything in state "Waiting for"
 - EVENT_NAME LIKE 'stage/%/Waiting for%'
- Frequently met issues

Stages Shortcuts

- Everything, related to temporary tables
- Everything, related to locks
- Everything in state "Waiting for"
- Frequently met issues
 - EVENT_NAME='stage/sql/freeing items'
 - EVENT_NAME='stage/sql/Sending data'
 - EVENT_NAME='stage/sql/cleaning up'
 - EVENT_NAME='stage/sql/closing tables'
 - EVENT_NAME='stage/sql/end'

Stages Example: Which Stage Run Critically Long?

```
mysql> SELECT eshl.event_name, sql_text, eshl.timer_wait/1000000000000 w_s
-> FROM performance_schema.events_stages_history_long eshl
-> JOIN performance_schema.events_statements_history_long esthl
-> ON (eshl.nesting_event_id = esthl.event_id)
-> WHERE eshl.timer_wait > 1*100000000000\G
***** 1. row *****
event_name: stage/sql/Sending data
sql_text: SELECT COUNT(emp_no) FROM employees JOIN salaries USING(emp_no)
          WHERE hire_date=from_date
w_s: 0.8170
1 row in set (0.00 sec)
```

5.7+: Prepared Statements



Table prepared_statements_instances

- Contains current prepared statements

Table prepared_statements_instances

- Contains current prepared statements
- Statistics by
 - Which thread owns the statement
 - How many times executed
 - Optimizer statistics, similar to events_statements_*

Prepared Statements: configuration

- Instruments

statement/sql/prepare_sql

statement/sql/execute_sql

statement/com/Prepare

statement/com/Execute

Prepared Statements: configuration

- Instruments
- Consumers
 - prepared_statements_instances

Prepared Statements: configuration

- Instruments
- Consumers
- Size

```
mysql> select @@performance_schema_max_prepared_statements_instances\G
***** 1. row *****
@@performance_schema_max_prepared_statements_instances: -1
1 row in set (0.00 sec)
```

Example: Prepared Statement

```
mysql1> prepare stmt from 'select count(*) from employees where hire_date > ?';
Query OK, 0 rows affected (0.00 sec)
Statement prepared
mysql1> set @hd='1995-01-01';
Query OK, 0 rows affected (0.00 sec)
mysql1> execute stmt using @hd;
+-----+
| count(*) |
+-----+
|      34004 |
+-----+
1 row in set (1.44 sec)
```

Example: Prepared Statement

```
mysql1> prepare stmt from 'select count(*) from employees where hire_date > ?';
Query OK, 0 rows affected (0.00 sec)
Statement prepared
mysql1> set @hd='1995-01-01';
Query OK, 0 rows affected (0.00 sec)
mysql1> execute stmt using @hd;
+-----+
| count(*) |
+-----+
|      34004 |
+-----+
1 row in set (1.44 sec)
```

- Try EXECUTE with different values

Example: diagnosis

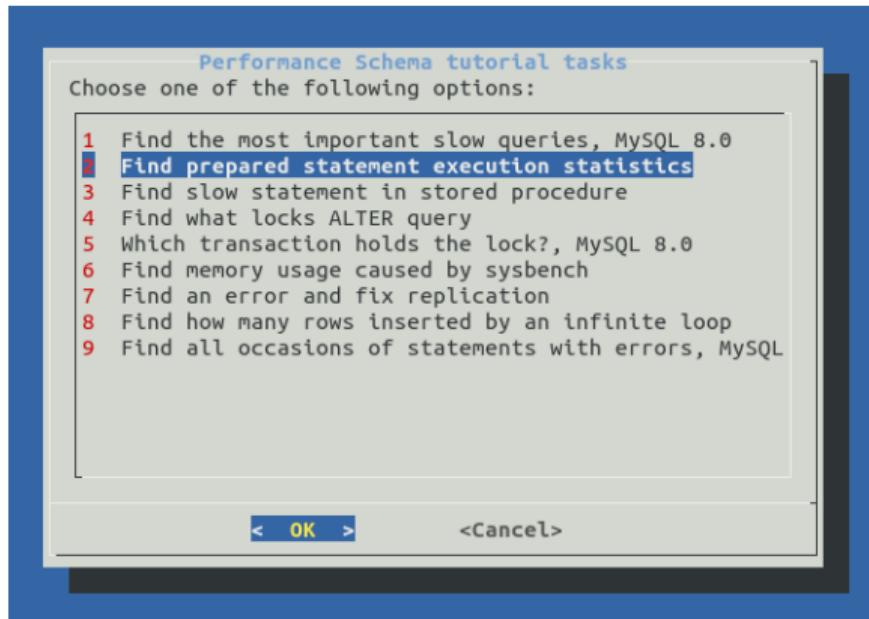
```
mysql2> select statement_name, sql_text, owner_thread_id, count_reprepare,
-> count_execute, sum_timer_execute from prepared_statements_instances\G
***** 1. row *****
statement_name: stmt
      sql_text: select count(*) from employees where hire_date > ?
owner_thread_id: 22
count_reprepare: 0
  count_execute: 3
sum_timer_execute: 4156561368000
1 row in set (0.00 sec)
```

```
mysql1> drop prepare stmt;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql2> select * from prepared_statements_instances\G
Empty set (0.00 sec)
```

Prepared Statements: practice

- Start the task



Prepared Statements: practice

- Inside MySQL client

```
CALL help_task()\G
```

```
CALL help_solve()\G
```

- We need to find out how effective is the prepared statement

5.7+: Stored Routines



New Instruments

```
mysql> select * from setup_instruments where name like 'statement/sp%';
+-----+-----+-----+
| NAME          | ENABLED | TIMED |
+-----+-----+-----+
| statement/sp/stmt      | YES     | YES    | | statement/sp/hreturn
| statement/sp/set       | YES     | YES    | | statement/sp/cpush
| statement/sp/set_trigger_field | YES     | YES    | | statement/sp/cpop
| statement/sp/jump      | YES     | YES    | | statement/sp/copen
| statement/sp/jump_if_not | YES     | YES    | | statement/sp/cclose
| statement/sp/freturn   | YES     | YES    | | statement/sp/cfetch
| statement/sp/hpush_jump | YES     | YES    | | statement/sp/error
| statement/sp/hpop      | YES     | YES    | | statement/sp/set_case_expr
| ...
+-----+-----+-----+
16 rows in set (0.00 sec)
```

Stored Routines Instrumentation

- What happens inside the routine

Stored Routines Instrumentation

- What happens inside the routine
- Queries, called from the routine
 - statement/sp/stmt

Stored Routines: example

- We will use this procedure

```
CREATE DEFINER='root'@'localhost' PROCEDURE 'sp_test'(val int)
BEGIN
    DECLARE CONTINUE HANDLER FOR 1364, 1048, 1366
    BEGIN
        INSERT IGNORE INTO t1 VALUES('Some string');
        GET STACKED DIAGNOSTICS CONDITION 1 @stacked_state = RETURNED_SQLSTATE;
        GET STACKED DIAGNOSTICS CONDITION 1 @stacked_msg = MESSAGE_TEXT;
    END;
    INSERT INTO t1 VALUES(val);
END
```

Stored Routines: example

- We will use this procedure

```
CREATE DEFINER='root'@'localhost' PROCEDURE 'sp_test'(val int)
BEGIN
    DECLARE CONTINUE HANDLER FOR 1364, 1048, 1366
    BEGIN
        INSERT IGNORE INTO t1 VALUES('Some string');
        GET STACKED DIAGNOSTICS CONDITION 1 @stacked_state = RETURNED_SQLSTATE;
        GET STACKED DIAGNOSTICS CONDITION 1 @stacked_msg = MESSAGE_TEXT;
    END;
    INSERT INTO t1 VALUES(val);
END
```

- When HANDLER called?

Correct Value

```
mysql> call sp_test(1);
Query OK, 1 row affected (0.07 sec)
```

```
mysql> select thread_id, event_name, sql_text from events_statements_history
-> where event_name like 'statement/sp%';
+-----+-----+-----+
| thread_id | event_name           | sql_text      |
+-----+-----+-----+
|     24 | statement/sp/hpush_jump | NULL          |
|     24 | statement/sp/stmt     | INSERT INTO t1 VALUES(val) |
|     24 | statement/sp/hpop     | NULL          |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

HANDLER call

```
mysql> call sp_test(NULL);
Query OK, 1 row affected (0.07 sec)
```

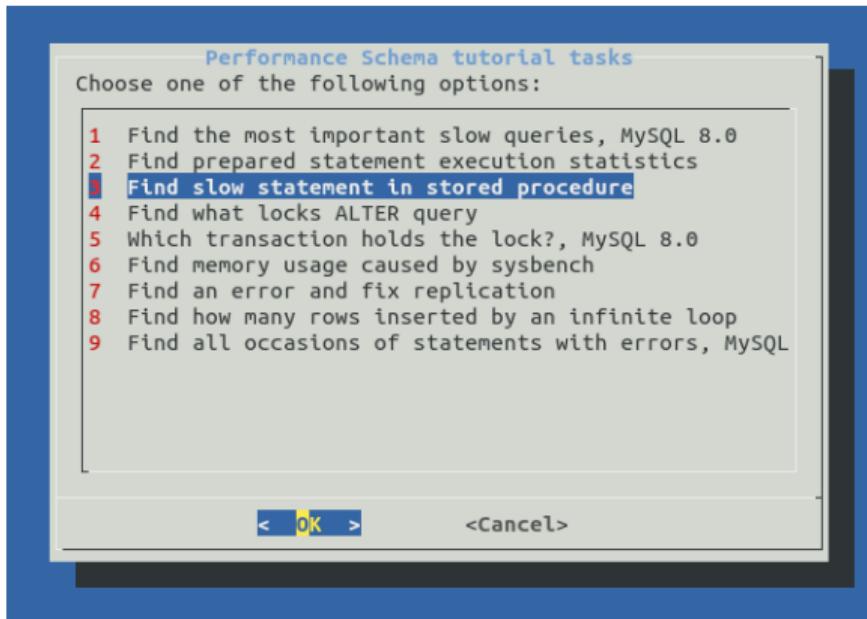
```
mysql> select thread_id, event_name, sql_text from events_statements_history
-> where event_name like 'statement/sp%';
```

thread_id	event_name	sql_text
24	statement/sp/hpush_jump	NULL
24	statement/sp/stmt	INSERT INTO t1 VALUES(val)
24	statement/sp/stmt	INSERT IGNORE INTO t1 VALUES('Some str...')
24	statement/sp/stmt	GET STACKED DIAGNOSTICS CONDITION 1 @s...
24	statement/sp/stmt	GET STACKED DIAGNOSTICS CONDITION 1 @s...
24	statement/sp/hreturn	NULL
24	statement/sp/hpop	NULL



Stored Routines: practice

- Start the task



Stored Routines: practice

- Inside MySQL client

```
CALL help_task()\G  
CALL help_solve()\G  
CALL task_prepare();
```

- We need to find out why procedure takes different time each run
- For better output set pager to less:

```
mysql> \P less
```

5.7+: Locks Diagnostic



5.7+: MDL

- Table METADATA_LOCKS

5.7+: MDL

- Table METADATA_LOCKS
- Which thread is waiting for a lock

5.7+: MDL

- Table METADATA_LOCKS
- Which thread is waiting for a lock
- Which thread holds the lock

5.7+: MDL

- Table METADATA_LOCKS
- Which thread is waiting for a lock
- Which thread holds the lock
- Not only for tables:

GLOBAL, SCHEMA, TABLE, FUNCTION, PROCEDURE, EVENT, COMMIT, USER LEVEL
LOCK, TABLESPACE

MDL: configuration

- Instruments
 - wait/lock/metadata/sql/mdl

MDL: configuration

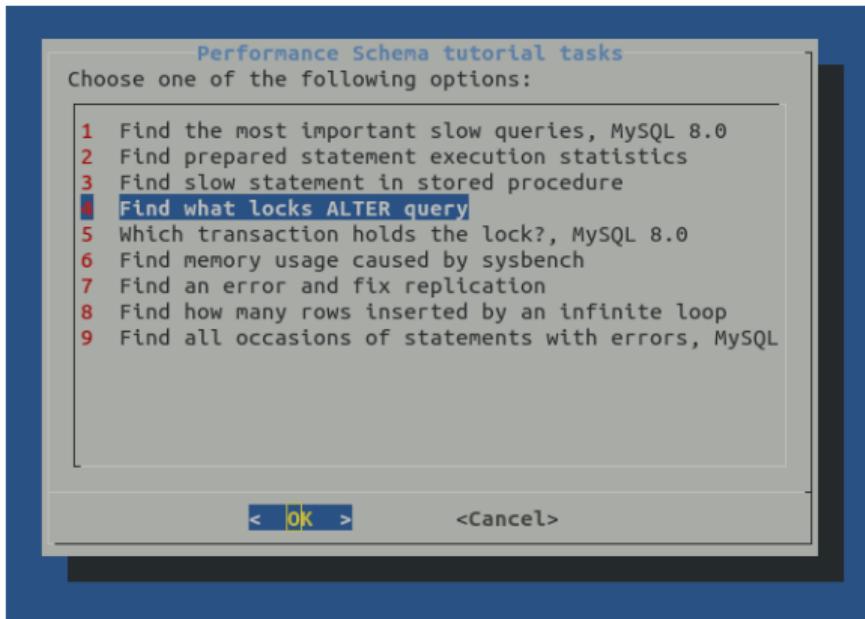
- Instruments
- Consumers
 - METADATA_LOCKS

METADATA_LOCKS: example

```
mysql> select processlist_id, object_type, lock_type, lock_status, source
-> from metadata_locks join threads on (owner_thread_id=thread_id)
-> where object_schema='employees' and object_name='titles'\G
***** 1. row *****
processlist_id: 4
  object_type: TABLE
  lock_type: EXCLUSIVE
  lock_status: PENDING -- waits
  source: mdl.cc:3263
***** 2. row *****
processlist_id: 5
  object_type: TABLE
  lock_type: SHARED_READ
  lock_status: GRANTED -- holds
  source: sql_parse.cc:5707
```

MDL: practice

- Start the task



MDL: practice

- Inside MySQL client

```
CALL help_task()\G  
CALL help_solve()\G  
CALL task_prepare();
```

- We need to find out what prevents ALTER from finishing

8.0.+: Data Locks

- Information about locks, held by engine

8.0.+: Data Locks

- Information about locks, held by engine
- Only for engines with own locking models

8.0.+: Data Locks

- Information about locks, held by engine
- Only for engines with own locking models
- Currently only InnoDB

8.0.+: Data Locks

- Information about locks, held by engine
- Only for engines with own locking models
- Currently only InnoDB
- Replacement for I_S tables
 - INNODB_LOCKS
 - INNODB_LOCK_WAITS

Data Locks: configuration

Unlike most Performance Schema data collection, there are no instruments for controlling whether data lock information is collected or system variables for controlling data lock table sizes. The Performance Schema collects information that is already available in the server, so there is no memory or CPU overhead to generate this information or need for parameters that control its collection.

<https://dev.mysql.com/doc/refman/8.0/en/data-locks-table.html>

Table DATA_LOCKS

- Which lock is held

```
***** 4. row *****  
    ENGINE: INNODB  
    ENGINE_LOCK_ID: 2408:0:393:2  
ENGINE_TRANSACTION_ID: 2408  
    THREAD_ID: 34  
    OBJECT_SCHEMA: test  
    OBJECT_NAME: t  
    INDEX_NAME: PRIMARY  
    LOCK_TYPE: RECORD  
    LOCK_MODE: X  
LOCK_STATUS: GRANTED  
    LOCK_DATA: 12345
```

Table DATA_LOCKS

- Which lock is held
- Which lock is requested

```
***** 2. row *****
```

```
    ENGINE: INNODB
    ENGINE_LOCK_ID: 2409:0:393:2
ENGINE_TRANSACTION_ID: 2409
    THREAD_ID: 36
    OBJECT_SCHEMA: test
    OBJECT_NAME: t
    INDEX_NAME: PRIMARY
    LOCK_TYPE: RECORD
    LOCK_MODE: X
LOCK_STATUS: WAITING
    LOCK_DATA: 12345
```

Table DATA_LOCKS

- Which lock is held
- Which lock is requested
- Both record-level and table level

```
p_s> select * from data_locks\G
*****
1. row *****

...
LOCK_TYPE: TABLE
LOCK_MODE: IX
LOCK_STATUS: GRANTED
LOCK_DATA: NULL
*****
2. row *****

...
LOCK_TYPE: RECORD
```

Table DATA_LOCK_WAIT

- Maps lock waits with granted locks

Table DATA_LOCK_WAITS

- Maps lock waits with granted locks
- Only granted blocking other transactions

```
p_s> select ENGINE, ... from data_lock_waits\G
***** 1. row *****
          ENGINE: INNODB
    REQUESTING_ENGINE_LOCK_ID: 2409:0:393:2
REQUESTING_ENGINE_TRANSACTION_ID: 2409
          REQUESTING_THREAD_ID: 36
    BLOCKING_ENGINE_LOCK_ID: 2408:0:393:2
BLOCKING_ENGINE_TRANSACTION_ID: 2408
          BLOCKING_THREAD_ID: 34
1 row in set (0,01 sec)
```

New Information

- Partition
- Subpartition
- Lock data
- Requesting and blocking thread id

In sys Schema

- View innodb_lock_waits

In sys Schema

- View innodb_lock_waits
- Takes additional information from INFORMATION_SCHEMA.INNODB_TRX

In sys Schema

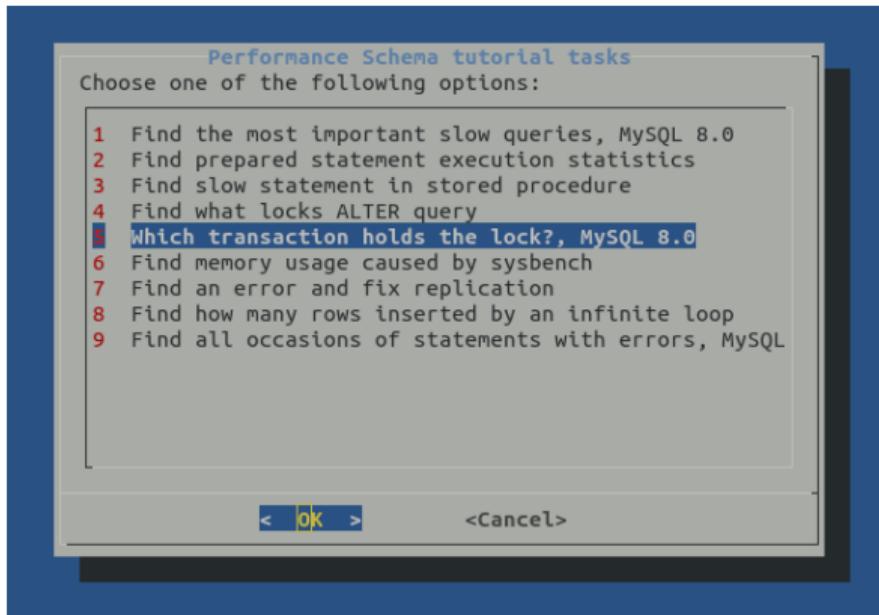
- View innodb_lock_waits

```
sys> select locked_table, ...
-> from innodb_lock_waits\G
***** 1. row *****
    locked_table: 'test'.'t'          blocking_pid: 4
    locked_index: PRIMARY           blocking_query: NULL
    locked_type: RECORD            blocking trx rows locked: 1
    waiting trx rows locked: 1    blocking trx rows modified: 1
    waiting trx rows modified: 0   sql kill blocking query: KILL QUERY 4
    waiting pid: 6                sql kill blocking connection: KILL 4

    waiting query: UPDATE t SET f='bar' WHERE id=12345
```

Data Locks: Practice

- Start the task



Data Locks: Practice

- Inside MySQL client

```
CALL help_task()\G
```

```
CALL help_solve()\G
```

- We need to find
 - Which transaction holds the lock
 - What is the missed statement
 - Which row is locked
 - Which partition is locked

5.7+: Memory Usage



Memory: configuration

• Instruments

```
mysql> select name from setup_instruments where name like 'memory%';
+-----+
| name |
+-----+
...
| memory/sql/Gtid_state::group_commit_sidno_locks |
| memory/sql/Mutex_cond_array::Mutex_cond           |
| memory/sql/TABLE_RULE_ENT                         |
| memory/sql/Rpl_info_table                        |
| memory/sql/Rpl_info_file::buffer                 |
| memory/sql/db_worker_hash_entry                  |
| memory/sql/rpl_slave::check_temp_dir            |
| memory/sql/rpl_slave::command_buffer            |
| memory/sql/binlog_ver_1_event                   |
+-----+
```

Memory: configuration

- Instruments
- Consumers
 - Digest tables in Performance Schema
 - Views in sys schema

Memory Diagnostic

- Memory, used by internal mysqld structures

Memory Diagnostic

- Memory, used by internal mysqld structures
- Aggregated by
 - Global
 - Thread
 - Account
 - Host
 - User

Memory Diagnostic

- Memory, used by internal mysqld structures
- Aggregated by
 - Global
 - Thread
 - Account
 - Host
 - User
- Nice views in sys schema

Memory Usage by Thread

```
mysql> select thread_id tid, user, current_allocated ca, total_allocated  
-> from sys.memory_by_thread_by_current_bytes;
```

tid	user	ca	total_allocated
1	sql/main	2.53 GiB	2.69 GiB
150	root@127.0.0.1	4.06 MiB	32.17 MiB
146	sql/slave_sql	1.31 MiB	1.44 MiB
145	sql/slave_io	1.08 MiB	2.79 MiB
...			
60	innodb/io_read_thread	0 bytes	384 bytes
139	innodb/srv_purge_thread	-328 bytes	754.21 KiB
69	innodb/io_write_thread	-1008 bytes	34.28 KiB
68	innodb/io_write_thread	-1440 bytes	298.05 KiB
74	innodb/io_write_thread	-1656 bytes	103.55 KiB
4	innodb/io_log_thread	-2880 bytes	132.38 KiB



Threads Statistics

```
mysql> select * from sys.memory_by_thread_by_current_bytes
    -> order by current_allocated desc\G
***** 1. row *****
    thread_id: 152
          user: lj@127.0.0.1
current_count_used: 325
  current_allocated: 36.00 GiB
  current_avg_alloc: 113.43 MiB
  current_max_alloc: 36.00 GiB
  total_allocated: 37.95 GiB
...
...
```

- Find threads, eating memory, in a second!

RAW Performance Schema tables

- `memory_summary_by_account_by_event_name`
- `memory_summary_by_host_by_event_name`
- `memory_summary_by_thread_by_event_name`
- `memory_summary_by_user_by_event_name`
- `memory_summary_global_by_event_name`

RAW Performance Schema tables

- `memory_summary_by_account_by_event_name`
- `memory_summary_by_host_by_event_name`
- `memory_summary_by_thread_by_event_name`
- `memory_summary_by_user_by_event_name`
- `memory_summary_global_by_event_name`
- You must enable memory instrumentation!

RAW Performance Schema tables

- `memory_summary_by_account_by_event_name`
- `memory_summary_by_host_by_event_name`
- `memory_summary_by_thread_by_event_name`
- `memory_summary_by_user_by_event_name`
- `memory_summary_global_by_event_name`
- **You must enable memory instrumentation!**
- sys schema includes user name

Users in sys.memory_* tables

- NAME@HOST - regular user

Users in sys.memory_* tables

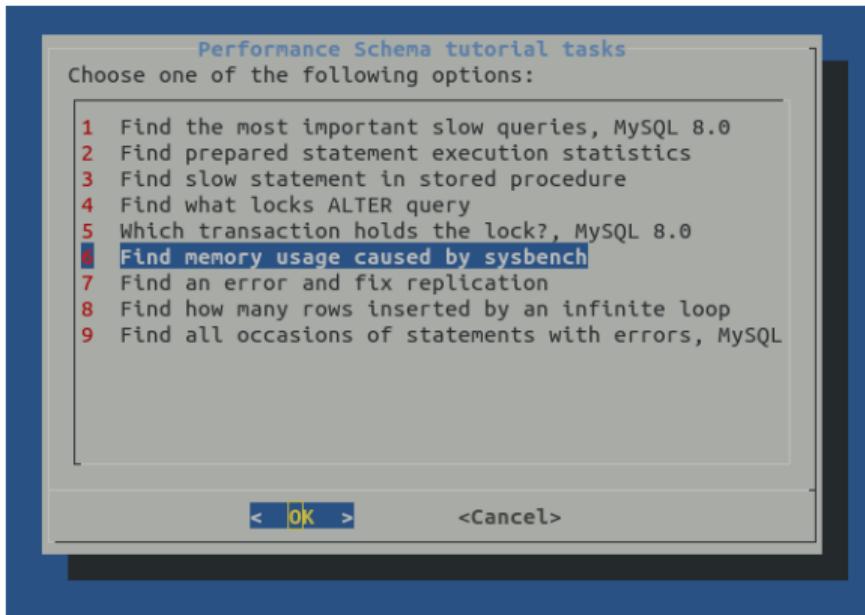
- NAME@HOST - regular user
- System users
 - sql/main
 - innodb/*
 - ...

Users in sys.memory_* tables

- NAME@HOST - regular user
- System users
 - sql/main
 - innodb/*
 - ...
- Data comes from table THREADS

Memory Usage: practice

- Start the task



Memory Usage: practice

- Inside MySQL client

```
CALL help_task()\G  
CALL help_solve()\G  
CALL task_prepare();
```

- We need to find out how much memory uses SysBench load, running in parallel
- To identify how much RAM used by whole server run

```
select * from sys.memory_global_total;
```

5.7+: Replication



Major Improvements

- Data from SHOW SLAVE STATUS available in replication_* tables

Major Improvements

- Data from SHOW SLAVE STATUS available in replication_* tables
- Support of Replication Channels (Multi-master slave)

Major Improvements

- Data from SHOW SLAVE STATUS available in replication_* tables
- Support of Replication Channels (Multi-master slave)
- More instruments for GTID

SLAVE STATUS

- No need to parse SHOW output

SLAVE STATUS

- No need to parse SHOW output
- Configuration
 - `replication_connection_configuration`
 - `replication_applier_configuration`

SLAVE STATUS

- No need to parse SHOW output
- Configuration
- IO thread
 - replication_connection_status

SLAVE STATUS

- No need to parse SHOW output
- Configuration
- IO thread
- SQL thread
 - replication_applier_status
 - replication_applier_status_by_coordinator - **MTS only**
 - replication_applier_status_by_worker
 - replication_applier_global_filters
 - replication_applier_filters

SLAVE STATUS

- No need to parse SHOW output
- Configuration
- IO thread
- SQL thread
- Group replication
 - `replication_group_members`
 - `replication_group_member_stats`

SLAVE STATUS

- Configuration

```
mysql> select * from replication_connection_configuration
    -> join replication_applier_configuration using(channel_name)\G
***** 1. row *****
CHANNEL_NAME:
    HOST: 127.0.0.1
    PORT: 13000
    USER: root
NETWORK_INTERFACE:
    AUTO_POSITION: 1
    SSL_ALLOWED: NO
    SSL_CA_FILE:
...
CHANNEL_NAME:
DESIRED_DELAY: 0
```

SLAVE STATUS

- State of IO Thread

```
mysql> select * from replication_connection_status\G
***** 1. row *****
CHANNEL_NAME:
GROUP_NAME:
SOURCE_UUID: d0753e78-14ec-11e5-b3fb-28b2bd7442fd
THREAD_ID: 21
SERVICE_STATE: ON
COUNT_RECEIVED_HEARTBEATS: 17
LAST_HEARTBEAT_TIMESTAMP: 2015-06-17 15:49:08
RECEIVED_TRANSACTION_SET:
    LAST_ERROR_NUMBER: 0
    LAST_ERROR_MESSAGE:
    LAST_ERROR_TIMESTAMP: 0000-00-00 00:00:00
1 row in set (0.00 sec)
```

Performance Schema: State of SQL Thread

- Coordinator thread for multiple workers

```
mysql> select * from replication_applier_status join
-> replication_applier_status_by_coordinator using(channel_name)\G
***** 1. row *****
      CHANNEL_NAME:
      SERVICE_STATE: ON
      REMAINING_DELAY: NULL
COUNT_TRANSACTIONS_RETRIES: 0
      THREAD_ID: 76
      SERVICE_STATE: ON
      LAST_ERROR_NUMBER: 0
      LAST_ERROR_MESSAGE:
      LAST_ERROR_TIMESTAMP: 0000-00-00 00:00:00.000000
LAST_PROCESSED_TRANSACTION: c634ce2d-cbb3-11e8-9532-0242cedce297
LAST_PROCESSED_TRANSACTION_ORIGINAL_COMMIT_TIMESTAMP: 2018-10-09
```

Performance Schema: State of SQL Thread

- Coordinator thread for multiple workers
- Multi-source slave

```
mysql> select * from replication_applier_status join
-> replication_applier_status_by_worker using(channel_name)\G
***** 1. row *****
      CHANNEL_NAME: master-01
      SERVICE_STATE: ON
      REMAINING_DELAY: NULL
...
LAST_APPLIED_TRANSACTION: c634ce2d-cbb3-11e8-9532-0242cedce297:7
LAST_APPLIED_TRANSACTION_ORIGINAL_COMMIT_TIMESTAMP: 2018-10-09 15:12:56.285995
...
```

Performance Schema: State of SQL Thread

- Coordinator thread for multiple workers
- Multi-source slave

```
***** 2. row *****  
    CHANNEL_NAME: m-03  
    SERVICE_STATE: OFF  
...  
    LAST_ERROR_NUMBER: 1050  
    LAST_ERROR_MESSAGE: Worker 1 failed executing transaction ...  
    LAST_ERROR_TIMESTAMP: 2018-10-09 15:15:14.306672  
LAST_APPLIED_TRANSACTION: c63ae70c-cbb3-11e8-b634-0242cedce297:2  
...  
LAST_APPLIED_TRANSACTION_END_APPLY_TIMESTAMP: 2018-10-09 15:13:33.770744  
    APPLYING_TRANSACTION: c63ae70c-cbb3-11e8-b634-0242cedce297:3  
...
```

GTID Diagnostics

- RECEIVED_TRANSACTION_SET
in table replication_connection_status

GTID Diagnostics

- RECEIVED_TRANSACTION_SET
in table replication_connection_status
- LAST_SEEN_TRANSACTION
in replication_applier_status_by_worker

GTID: All in One Place

- Single-threaded slave

```
mysql> select cs.CHANNEL_NAME, cs.SOURCE_UUID, cs.RECEIVED_TRANSACTION_SET,
-> asw.LAST_APPLIED_TRANSACTION, aps.SERVICE_STATE from
-> replication_connection_status cs join replication_applier_status_by_work
-> asw using(channel_name) join replication_applier_status aps
-> using(channel_name) \G
*****
          1. row *****
CHANNEL_NAME:
      SOURCE_UUID: 9038967d-7164-11e6-8c88-30b5c2208a0f
RECEIVED_TRANSACTION_SET: 9038967d-7164-11e6-8c88-30b5c2208a0f:1-2
LAST_APPLIED_TRANSACTION: 9038967d-7164-11e6-8c88-30b5c2208a0f:2
      SERVICE_STATE: ON
1 row in set (0,00 sec)
```

GTID: All in One Place

- Single-threaded slave
- Multi-threaded

```
***** 1. row *****
      THREAD_ID: 30
      SERVICE_STATE: ON
RECEIVED_TRANSACTION_SET: 9038967d-7164-11e6-8c88-30b5c2208a0f:1-3
LAST_APPLIED_TRANSACTION:
...
***** 8. row *****
      THREAD_ID: 37
      SERVICE_STATE: ON
RECEIVED_TRANSACTION_SET: 9038967d-7164-11e6-8c88-30b5c2208a0f:1-3
LAST_APPLIED_TRANSACTION: 9038967d-7164-11e6-8c88-30b5c2208a0f:3
8 rows in set (0,00 sec)
```



Replication Filters

- Global

```
mysql> select * from replication_applier_global_filters\G
***** 1. row *****
  FILTER_NAME: REPLICATE_IGNORE_TABLE
  FILTER_RULE: test.test
CONFIGURED_BY: CHANGE_REPLICATION_FILTER
 ACTIVE_SINCE: 2018-10-09 16:54:45.733490
1 row in set (0.00 sec)
```

Replication Filters

- Global
- Worker's

```
mysql> select * from replication_applier_filters\G
***** 1. row *****
CHANNEL_NAME: m-01
FILTER_NAME: REPLICATE_IGNORE_TABLE
FILTER_RULE: test.test
CONFIGURED_BY: CHANGE_REPLICATION_FILTER
ACTIVE_SINCE: 2018-10-09 16:54:45.733490
COUNTER: 0
***** 2. row *****
CHANNEL_NAME: m-03
FILTER_NAME: REPLICATE_IGNORE_TABLE
FILTER_RULE: test.t1
```

Group Replication

- `replication_group_members`

```
mysql> SELECT * FROM performance_schema.replication_group_members\G
***** 1. row *****
CHANNEL_NAME: group_replication_applier
    MEMBER_ID: c26fc705-cbcc-11e8-b5b8-0242cedce297
    MEMBER_HOST: delly
    MEMBER_PORT: 24801
    MEMBER_STATE: ONLINE
    MEMBER_ROLE: PRIMARY
MEMBER_VERSION: 8.0.12
...
```

Group Replication

- replication_group_members

```
***** 2. row *****
```

```
CHANNEL_NAME: group_replication_applier
```

```
 MEMBER_ID: f3d2feda-cbcc-11e8-9eb9-0242cedce297
```

```
 MEMBER_HOST: delly
```

```
 MEMBER_PORT: 24802
```

```
 MEMBER_STATE: RECOVERING
```

```
 MEMBER_ROLE: PRIMARY
```

```
 MEMBER_VERSION: 8.0.12
```

```
2 rows in set (0.00 sec)
```

Group Replication

- replication_group_members
- replication_group_member_stats

```
mysql> SELECT * FROM performance_schema.replication_group_member_stats\G
***** 1. row *****
      CHANNEL_NAME: group_replication_applier
      VIEW_ID: 15390963914656312:3
      MEMBER_ID: 1dfa4dee-cbcd-11e8-bad1-0242cedce297
      COUNT_TRANSACTIONS_IN_QUEUE: 0
      COUNT_TRANSACTIONS_CHECKED: 1
      COUNT_CONFLICTS_DETECTED: 0
      COUNT_TRANSACTIONS_ROWS_VALIDATING: 8
      ...
```

More Diagnostic

- Tables in mysql schema
 - slave_master_info
 - slave_relay_log_info
 - slave_worker_info
 - Join with Performance Schema tables

More Diagnostic

- Tables in mysql schema
 - slave_master_info
 - slave_relay_log_info
 - slave_worker_info
 - Join with Performance Schema tables
- New instruments
 - memory
 - wait
 - stage

Table log_status

- Not only replication

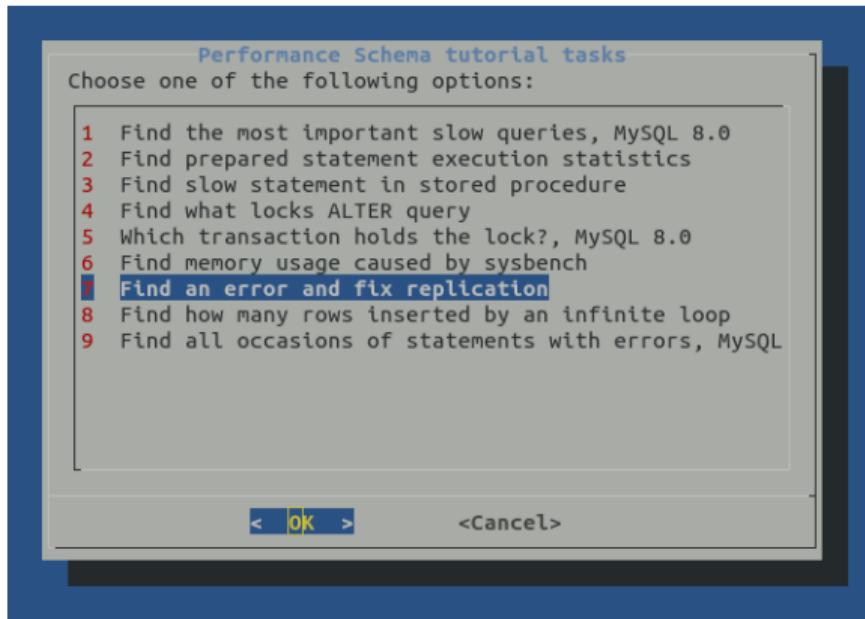
```
mysql> select * from log_status\G
***** 1. row *****
 SERVER_UUID: c634ce2c-cbb3-11e8-8900-0242cedce297
      LOCAL: {"gtid_executed": "c634ce2d-cbb3-11e8-9532-0242cedce297:1-2",
                "binary_log_file": "binlog.000001",
                "binary_log_position": 620}
REPLICATION: {"channels": [{"channel_name": "",
                            "relay_log_file": "delly-relay-bin.000002",
                            "relay_log_position": 819,
                            "relay_master_log_file": "binlog.000001",
                            "exec_master_log_position": 611}]}
STORAGE_ENGINES: {"InnoDB": {"LSN": 19177930, "LSN_checkpoint": 19177930}}
1 row in set (0.01 sec)
```

Table log_status

- Not only replication
- Currently only InnoDB supported!

Replication: practice

- Start the task



Replication: practice

- Inside MySQL client

```
CALL help_task()\G
```

```
CALL help_solve()\G
```

- To open another terminal: Ctrl+b , c
- Connection commands in another terminal
 - Master: m
 - Slave: s1
- We need to find out why replication is broken and fix it

5.7+: Variables



Variables Instrumentation

- Variables

- global_variables
- session_variables
- user_variables_by_thread
- variables_by_thread

Variables Instrumentation

- Variables
- Status variables
 - global_status
 - session_status
 - status_by_[account|host|thread|user]

Variables Instrumentation

- Variables
- Status variables
- `show_compatibility_56 = 0`

Global and Session Variables

- Same information which is in

- `SHOW [GLOBAL] STATUS`
- `I_S.GLOBAL_VARIABLES`

- Deprecated in 5.7
- Removed in 8.0.1

- `I_S.SESSION_VARIABLES`

- Deprecated in 5.7
- Removed in 8.0.1

Global and Session Variables

- Same information which is in
 - SHOW [GLOBAL] STATUS
 - I_S.GLOBAL_VARIABLES
 - Deprecated in 5.7
 - Removed in 8.0.1
 - I_S.SESSION_VARIABLES
 - Deprecated in 5.7
 - Removed in 8.0.1
- Helps to watch session variables changes

Status Variables

- Same information which is in

- `SHOW [GLOBAL] STATUS`
- `I_S.GLOBAL_STATUS`

- Deprecated in 5.7
- Removed in 8.0.1

- `I_S.SESSION_STATUS`

- Deprecated in 5.7
- Removed in 8.0.1

Status Variables

```
mysql> SELECT ss.variable_name, ss.variable_value FROM session_status ss
    -> LEFT JOIN global_status gs USING(variable_name)
    -> WHERE ss.variable_value != gs.variable_value OR gs.variable_value IS NULL;
+-----+-----+
| variable_name          | variable_value |
+-----+-----+
| Bytes_sent              | 197774          |
| Handler_commit           | 0               |
| Handler_external_lock   | 44              |
| Handler_read_first       | 3               |
| Handler_read_key         | 523             |
| Handler_read_next         | 0               |
| Handler_read_rnd_next    | 7241            |
| Opened_table_definitions | 0               |
...

```

Possible to Group

- variables_by_thread
- status_by_
 - account
 - host
 - thread
 - user

Possible to Group

- variables_by_thread

```
mysql> select * from variables_by_thread where variable_name='tx_isolation';
+-----+-----+-----+
| THREAD_ID | VARIABLE_NAME | VARIABLE_VALUE |
+-----+-----+-----+
|      71 | tx_isolation | REPEATABLE-READ |
|      83 | tx_isolation | REPEATABLE-READ |
|      84 | tx_isolation | SERIALIZABLE   |
+-----+-----+-----+
3 rows in set, 3 warnings (0.00 sec)
```

Possible to Group

- variables_by_thread
- status_by_

```
mysql> select * from status_by_thread where variable_name='Handler_write';
+-----+-----+-----+
| THREAD_ID | VARIABLE_NAME | VARIABLE_VALUE |
+-----+-----+-----+
|        71 | Handler_write |      94          |
|       83 | Handler_write |     477          | -- Most writes
|       84 | Handler_write |      101          |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

User Variables

- Grouped by connection
- Sometimes can help to find tricky bugs with persistent connections

```
mysql> select * from user_variables_by_thread;
+-----+-----+-----+
| THREAD_ID | VARIABLE_NAME | VARIABLE_VALUE |
+-----+-----+-----+
|      71 | baz          | boo           |
|     84 | foo          | bar           |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

8.0+: Variables Info

- **VARIABLES_INFO**
 - Source of variable
 - COMPILED
 - EXPLICIT
 - COMMAND_LINE
 - DYNAMIC
 - Path of option file if specified
 - Minimum and maximum values

8.0+: Variables Info

● VARIABLES_INFO

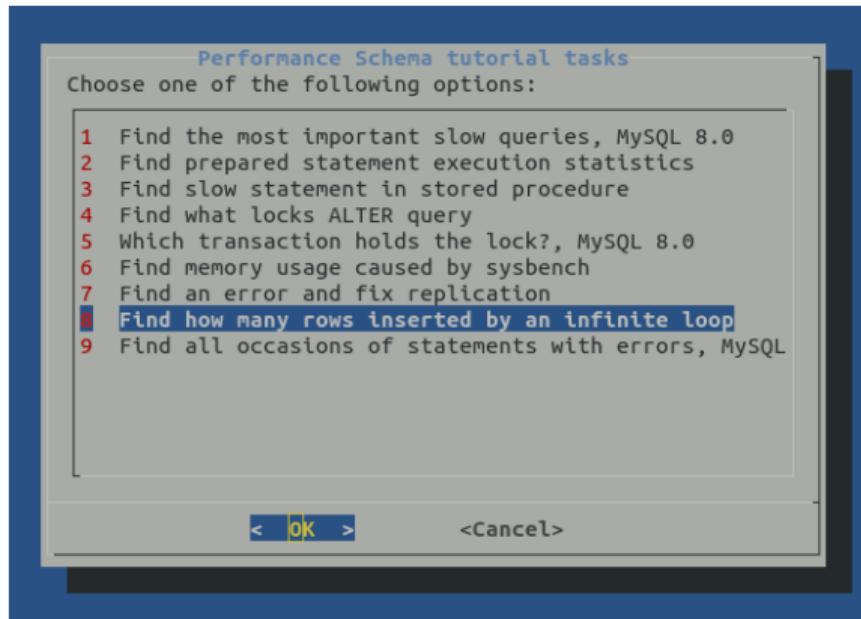
```
mysql> select * from variables_info where set_user is not null\G
***** 1. row *****
VARIABLE_NAME: innodb_file_per_table
VARIABLE_SOURCE: DYNAMIC
VARIABLE_PATH:
    MIN_VALUE: 0
    MAX_VALUE: 0
    SET_TIME: 2018-10-09 22:17:34.638459
    SET_USER: root
    SET_HOST: localhost
***** 2. row *****
VARIABLE_NAME: max_allowed_packet
VARIABLE_SOURCE: DYNAMIC
VARIABLE_PATH:
```

8.0+: Variables Info

- VARIABLES_INFO
 - Source of variable
 - COMPILED
 - EXPLICIT
 - COMMAND_LINE
 - DYNAMIC
 - Path of option file if specified
 - Minimum and maximum values
- No variable values in this table!

Variables: practice

- Start the task



Variables: practice

- Inside MySQL client

```
CALL help_task()\G  
CALL help_solve()\G  
CALL task_prepare();
```

- We need to watch progress of `INSERT` command, running by stored routine.
- Note what there is parallel load, caused by SysBench. We are not interested in its statistics.

8.0+: Errors Summary



Errors Summary Tables

- Traditionally aggregated
 - events_errors_summary_by_account_by_error
 - events_errors_summary_by_host_by_error
 - events_errors_summary_by_thread_by_error
 - events_errors_summary_by_user_by_error
 - events_errors_summary_global_by_error

Errors Summary Tables

- Traditionally aggregated
- All tables have similar structure

```
mysql> DESC events_errors_summary_global_by_error;
```

Field	Type	Null	Key	Default
ERROR_NUMBER	int(11)	YES	UNI	NULL
ERROR_NAME	varchar(64)	YES		NULL
SQL_STATE	varchar(5)	YES		NULL
SUM_ERROR_RAISED	bigint(20) unsigned	NO		NULL
SUM_ERROR_HANDLED	bigint(20) unsigned	NO		NULL
FIRST_SEEN	timestamp	YES		0000-00-00 00:00:00
LAST_SEEN	timestamp	YES		0000-00-

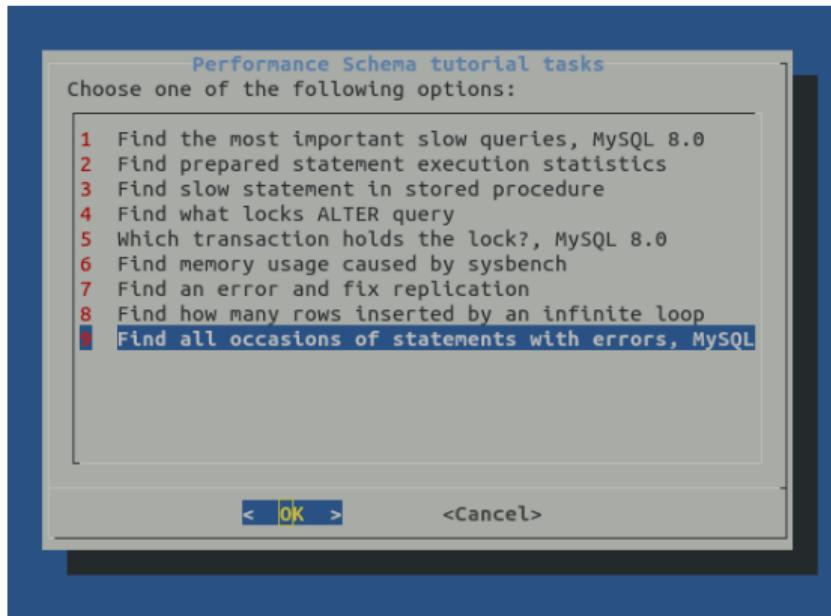
Errors Summary: Which Accounts Raise More Errors?

```
mysql> select * from events_errors_summary_by_account_by_error  
-> where SUM_ERROR_RAISED > 100\G
```

***** 1. row *****		***** 2. row *****	
USER:	root	USER:	root
HOST:	localhost	HOST:	localhost
ERROR_NUMBER:	1213	ERROR_NUMBER:	1287
ERROR_NAME:	ER_LOCK_DEADLOCK	ERROR_NAME:	ER_WARN_DEPRECATED_SYNTAX
SQL_STATE:	40001	SQL_STATE:	HY000
SUM_ERROR_RAISED:	221	SUM_ERROR_RAISED:	279
SUM_ERROR_HANDED:	0	SUM_ERROR_HANDED:	0
FIRST_SEEN:	2016-09-28 01:45:09	FIRST_SEEN:	2016-09-27 23:59:49
LAST_SEEN:	2016-09-28 01:47:02	LAST_SEEN:	2016-09-28 01:47:02

Errors Summary: practice

- Start the task



Errors Summary: practice

- Inside MySQL client

```
CALL help_task()\G
```

```
CALL help_solve()\G
```

- We need to find all occasions of statements with errors
 - Get the list all accounts which raised errors
 - Get the top 3 error types globally

Practice: Download Code



github.com/svetasmirnova/performance_schema_tutorial

More information



Blog of MySQL developers team



Mark Leith: author of sys schema



Official reference manual



Webinar "Performance Schema for MySQL Troubleshooting"

Rate Our Session!

The image displays three screenshots of a mobile application interface, likely from the Percona Live app, showing a schedule, session details, and a rating screen.

Schedule Screen: Shows a weekly schedule from Monday to Friday. A green callout box labeled "TAP THE SESSION" points to a specific session: "MySQL Performance Schema in Action: the Complete Tutorial" on November 05, 09:00 - 12:00, Wall Street 2+3. This session has no ratings yet.

Session Details Screen: Provides more details about the selected session. It includes the title, date, time, location, and a brief description: "Performance Schema is powerful diagnostic instrument for: - Query performance - Complicated locking issues - Memory leaks - Resource usage - Problematic behavior, caused by inappropriate settings - More It comes with hundreds of options which allow precisely tune what to Instrument. More". It also lists speakers: Alexander Rubin (Principal Architect (Consulting) at Percona) and Sveta Smirnova (Principal Support Escalation Specialist at Percona).

Rate & Review Screen: A rating interface with the heading "Rate & Review". It features a "Tap a star to rate" section with five stars, a "Review (optional)" input field, an "Anonymously" toggle switch, and a large blue "Submit" button.



PERCONA
LIVE

Thank you!

<http://www.slideshare.net/SvetaSmirnova>

<https://twitter.com/svetasmirnova>

<https://github.com/svetasmirnova>

<https://www.linkedin.com/in/alexanderrubin>