Create, Delete, and Alter a Stored Procedure

This lesson covers the basics on creating, viewing, altering and removing a stored procedure.

We'll cover the following

Create, Alter, and Delete a Stored Procedure

To create a stored procedure we use the **CREATE PROCEDURE** statement. **CREATE PROCEDURE** keywords are followed by the procedure name. We can also specify parameters to our procedure as a comma separated list withing parenthesis after the procedure name. The body of the stored procedure is enclosed with in the **BEGIN** and **END** keywords.

In order to delete a stored procedure, the **ALTER ROUTINE** privilege is a must. The **DROP PROCEDURE** statement deletes a stored procedure from the database. It can be used with the optional **IF EXISTS** clause.

Syntax

```
DELIMITER **

CREATE PROCEDURE procedure_name( parameter_list )

BEGIN

procedure_body

END**
```

DELIMITER ;

DROP PROCEDURE [IF EXISTS] **procedure_name**;

Connect to the terminal below by clicking in the widget. Once connected, the command line prompt will show up. Enter or copy-paste the command **./DataJek/Lessons/51lesson.sh** and wait for the mysql prompt to start-up.

```
-- The lesson queries are reproduced below for convenient copy/paste into the terminal.

-- Query 1
DELIMITER **
CREATE PROCEDURE ShowActors()
BEGIN
    SELECT *  FROM Actors;
END **
DELIMITER ;

-- Query 2
CALL ShowActors();

-- Query 3
SHOW PROCEDURE STATUS;

-- Query 4
SHOW PROCEDURE STATUS WHERE db = 'MovieIndustry';

-- Query 5
SELECT routine_name
FROM information_schema.routines
WHERE routine_type = 'PROCEDURE'
    AND routine_schema = 'sys';

-- Query 6
DROP PROCEDURE IF EXISTS ShowActors;
```

Terminal

1. We can create a stored procedure **ShowActors** that displays all the actors in our database as follows:

```
DELIMITER **
CREATE PROCEDURE ShowActors()
BEGIN
    SELECT *  FROM Actors;
```

```
END **
DELIMITER ;
```

A stored procedure named **ShowActors** is created which, when called, will display all the actors. We have used the **DELIMITER** command in line 1. By default semicolon (;) is used to separate two statements. Since a stored procedure can have multiple statements that end with semicolon character, it will not be considered as a single statement. The **DELIMITER** keyword is used to redefine the delimiter to ** so that we can pass the whole stored procedure to the server as a single statement. We use our redefined delimiter in line 5 to signal the end of the stored procedure. Then the delimiter is set back to semicolon in the last line.

The body of our stored procedure consists of a simple **SELECT** statement.

2. The next step is to execute our stored procedure. The **CALL** statement is used to invoke a stored procedure as follows:

```
CALL ShowActors();
```

Executing the above statement is the same as executing a query. The image shows the results of ShowActors procedure:

3. To view the stored procedures in a database, we will use the **SHOW PROCEDURE STATUS** statement as follows:

```
SHOW PROCEDURE STATUS;
```

The output of the above command is copious and captured in the widget below:

```
# OUPUT OF SHOW PROCEDURE;

mysql> SHOW PROCEDURE STATUS;
+---------------+-----------------------------------+-----------+---------------------+---------
| Db            | Name                              | Type      | Definer             | Modified
+---------------+-----------------------------------+-----------+---------------------+---------
| MovieIndustry | ShowActors                        | PROCEDURE | root@localhost      | 2020-06-
| sys           | create_synonym_db                 | PROCEDURE | mysql.sys@localhost | 2020-06-
 Description
```

```
Takes a source database name and synonym name, and then creates the
synonym database with views that point to all of the tables within

the source database.

Useful for creating a "ps" synonym for "performance_schema",
or "is" instead of "information_schema", for example.

Parameters

in_db_name (VARCHAR(64)):
The database name that you would like to create a synonym for.
in_synonym (VARCHAR(64)):
The database synonym name.

Example

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| test               |
+--------------------+
5 rows in set (0.00 sec)

mysql> CALL sys.create_synonym_db('performance_schema', 'ps');
+---------------------------------------+
| summary                               |
+---------------------------------------+
| Created 74 views in the `ps` database |
+---------------------------------------+
1 row in set (8.57 sec)

Query OK, 0 rows affected (8.57 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| ps                 |
| sys                |
| test               |
+--------------------+
6 rows in set (0.00 sec)

mysql> SHOW FULL TABLES FROM ps;
+---------------------------------------------------+------------+
| Tables_in_ps                                      | Table_type |
+---------------------------------------------------+------------+
| accounts                                          | VIEW       |
| cond_instances                                    | VIEW       |
| events_stages_current                             | VIEW       |
| events_stages_history                             | VIEW       |
...
 | utf8                | utf8_general_ci      | utf8_general_ci    |
```

```
| sys           | diagnostics                          | PROCEDURE | mysql.sys@localhost | 2020-06-0
 Description

 Create a report of the current status of the server for diagnostics purposes. Data collected inclu

 * The GLOBAL VARIABLES
 * Several sys schema views including metrics or equivalent (depending on version and settings)
 * Queries in the 95th percentile
 * Several ndbinfo views for MySQL Cluster
 * Replication (both master and slave) information.

 Some of the sys schema views are calculated as initial (optional), overall, delta:

 * The initial view is the content of the view at the start of this procedure.
 This output will be the same as the the start values used for the delta view.
 The initial view is included if @sys.diagnostics.include_raw = 'ON'.
 * The overall view is the content of the view at the end of this procedure.
 This output is the same as the end values used for the delta view.
 The overall view is always included.
 * The delta view is the difference from the beginning to the end. Note that for min and max value
 they are simply the min or max value from the end view respectively, so does not necessarily refl
 the minimum/maximum value in the monitored period.
 Note: except for the metrics views the delta is only calculation between the first and last output

 Requires the SUPER privilege for "SET sql_log_bin = 0;".

 Versions supported:
 * MySQL 5.6: 5.6.10 and later
 * MySQL 5.7: 5.7.9 and later

 Parameters

 in_max_runtime (INT UNSIGNED):
 The maximum time to keep collecting data.
 Use NULL to get the default which is 60 seconds, otherwise enter a value greater than 0.
 in_interval (INT UNSIGNED):
 How long to sleep between data collections.
 Use NULL to get the default which is 30 seconds, otherwise enter a value greater than 0.
 in_auto_config (ENUM('current', 'medium', 'full'))
 Automatically enable Performance Schema instruments and consumers.
 NOTE: The more that are enabled, the more impact on the performance.
 Supported values are:
 * current - use the current settings.
 * medium - enable some settings.
 * full - enables all settings. This will have a big impact on the
 performance - be careful using this option.
 If another setting the 'current' is chosen, the current settings
 are restored at the end of the procedure.


 Configuration Options

 sys.diagnostics.allow_i_s_tables
 Specifies whether it is allowed to do table scan queries on information_schema.TABLES. This can be
 are many tables. Set to 'ON' to allow, 'OFF' to not allow.
 Default is 'OFF'.

 sys.diagnostics.include_raw
 Set to 'ON' to include the raw data (e.g. the original output of "SELECT * FROM sys.metrics").
 Use this to get the initial values of the various views.
 Default is 'OFF'.
```

```
sys.statement_truncate_len
How much of queries in the process list output to include.
Default is 64.

sys.debug
Whether to provide debugging output.
Default is 'OFF'. Set to 'ON' to include.


Example

To create a report and append it to the file diag.out:

mysql> TEE diag.out;
mysql> CALL sys.diagnostics(120, 30, 'current');
...
mysql> NOTEE;
 | utf8                 | utf8_general_ci      | utf8_general_ci    |
| sys           | execute_prepared_stmt               | PROCEDURE | mysql.sys@localhost | 2020-06-
Description

Takes the query in the argument and executes it using a prepared statement. The prepared statement
so the procedure is mainly useful for executing one off dynamically created queries.

The sys_execute_prepared_stmt prepared statement name is used for the query and is required not to


Parameters

in_query (longtext CHARACTER SET UTF8):
The query to execute.


Configuration Options

sys.debug
Whether to provide debugging output.
Default is 'OFF'. Set to 'ON' to include.


Example

mysql> CALL sys.execute_prepared_stmt('SELECT * FROM sys.sys_config');
+------------------------+-------+--------------------+--------+
| variable               | value | set_time           | set_by |
+------------------------+-------+--------------------+--------+
| statement_truncate_len | 64    | 2015-06-30 13:06:00 | NULL   |
+------------------------+-------+--------------------+--------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
       | utf8                 | utf8_general_ci      | utf8_general_ci    |
| sys           | ps_setup_disable_background_threads | PROCEDURE | mysql.sys@localhost | 2020-06-
Description

Disable all background thread instrumentation within Performance Schema.

Parameters

None.

Example
```

```
mysql> CALL sys.ps_setup_disable_background_threads();
+--------------------------------+
| summary                        |
+--------------------------------+
| Disabled 18 background threads |
+--------------------------------+
1 row in set (0.00 sec)

| sys           | ps_setup_disable_consumer           | PROCEDURE | mysql.sys@localhost | 2020-06-0
Description

Disables consumers within Performance Schema
matching the input pattern.

Parameters

consumer (VARCHAR(128)):
A LIKE pattern match (using "%consumer%") of consumers to disable

Example

To disable all consumers:

mysql> CALL sys.ps_setup_disable_consumer('');
+--------------------------+
| summary                  |
+--------------------------+
| Disabled 15 consumers    |
+--------------------------+
1 row in set (0.02 sec)

To disable just the event_stage consumers:

mysql> CALL sys.ps_setup_disable_comsumers('stage');
+------------------------+
| summary                |
+------------------------+
| Disabled 3 consumers   |
+------------------------+
1 row in set (0.00 sec)

| sys           | ps_setup_disable_instrument         | PROCEDURE | mysql.sys@localhost | 2020-06-0
Description

Disables instruments within Performance Schema
matching the input pattern.

Parameters

in_pattern (VARCHAR(128)):
A LIKE pattern match (using "%in_pattern%") of events to disable

Example

To disable all mutex instruments:

mysql> CALL sys.ps_setup_disable_instrument('wait/synch/mutex');
+--------------------------+
| summary                  |
+--------------------------+
| Disabled 155 instruments |
```

```
+--------------------------+
1 row in set (0.02 sec)

To disable just a specific TCP/IP based network IO instrument:

mysql> CALL sys.ps_setup_disable_instrument('wait/io/socket/sql/server_tcpip_socket');
+------------------------+
| summary                |
+------------------------+
| Disabled 1 instruments |
+------------------------+
1 row in set (0.00 sec)

To disable all instruments:

mysql> CALL sys.ps_setup_disable_instrument('');
+--------------------------+
| summary                  |
+--------------------------+
| Disabled 547 instruments |
+--------------------------+
1 row in set (0.01 sec)
  | utf8                 | utf8_general_ci      | utf8_general_ci    |
| sys           | ps_setup_disable_thread             | PROCEDURE | mysql.sys@localhost | 2020-06-
Description

Disable the given connection/thread in Performance Schema.

Parameters

in_connection_id (BIGINT):
The connection ID (PROCESSLIST_ID from performance_schema.threads
or the ID shown within SHOW PROCESSLIST)

Example

mysql> CALL sys.ps_setup_disable_thread(3);
+-------------------+
| summary           |
+-------------------+
| Disabled 1 thread |
+-------------------+
1 row in set (0.01 sec)

To disable the current connection:

mysql> CALL sys.ps_setup_disable_thread(CONNECTION_ID());
+-------------------+
| summary           |
+-------------------+
| Disabled 1 thread |
+-------------------+
1 row in set (0.00 sec)

| sys           | ps_setup_enable_background_threads  | PROCEDURE | mysql.sys@localhost | 2020-06-
Description

Enable all background thread instrumentation within Performance Schema.

Parameters

None.
```

```
 Example

 mysql> CALL sys.ps_setup_enable_background_threads();
 +-------------------------------+
 | summary                       |
 +-------------------------------+
 | Enabled 18 background threads |
 +-------------------------------+
 1 row in set (0.00 sec)

| sys           | ps_setup_enable_consumer            | PROCEDURE | mysql.sys@localhost | 2020-06-
 Description

 Enables consumers within Performance Schema
 matching the input pattern.

 Parameters

 consumer (VARCHAR(128)):
 A LIKE pattern match (using "%consumer%") of consumers to enable

 Example

 To enable all consumers:

 mysql> CALL sys.ps_setup_enable_consumer('');
 +-------------------------+
 | summary                 |
 +-------------------------+
 | Enabled 10 consumers    |
 +-------------------------+
 1 row in set (0.02 sec)

 Query OK, 0 rows affected (0.02 sec)

 To enable just "waits" consumers:

 mysql> CALL sys.ps_setup_enable_consumer('waits');
 +----------------------+
 | summary              |
 +----------------------+
 | Enabled 3 consumers  |
 +----------------------+
 1 row in set (0.00 sec)

 Query OK, 0 rows affected (0.00 sec)

| sys           | ps_setup_enable_instrument          | PROCEDURE | mysql.sys@localhost | 2020-06-
 Description

 Enables instruments within Performance Schema
 matching the input pattern.

 Parameters

 in_pattern (VARCHAR(128)):
 A LIKE pattern match (using "%in_pattern%") of events to enable

 Example

 To enable all mutex instruments:
```

```
mysql> CALL sys.ps_setup_enable_instrument('wait/synch/mutex');
+------------------------+
| summary                |
+------------------------+
| Enabled 155 instruments |
+------------------------+
1 row in set (0.02 sec)

Query OK, 0 rows affected (0.02 sec)

To enable just a specific TCP/IP based network IO instrument:

mysql> CALL sys.ps_setup_enable_instrument('wait/io/socket/sql/server_tcpip_socket');
+----------------------+
| summary              |
+----------------------+
| Enabled 1 instruments |
+----------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

To enable all instruments:

mysql> CALL sys.ps_setup_enable_instrument('');
+-------------------------+
| summary                 |
+-------------------------+
| Enabled 547 instruments |
+-------------------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)
  | utf8                 | utf8_general_ci      | utf8_general_ci    |
| sys           | ps_setup_enable_thread              | PROCEDURE | mysql.sys@localhost | 2020-06-
Description

Enable the given connection/thread in Performance Schema.

Parameters

in_connection_id (BIGINT):
The connection ID (PROCESSLIST_ID from performance_schema.threads
or the ID shown within SHOW PROCESSLIST)

Example

mysql> CALL sys.ps_setup_enable_thread(3);
+------------------+
| summary          |
+------------------+
| Enabled 1 thread |
+------------------+
1 row in set (0.01 sec)

To enable the current connection:

mysql> CALL sys.ps_setup_enable_thread(CONNECTION_ID());
+------------------+
| summary          |
+------------------+
```

```
| Enabled 1 thread |
+------------------+
1 row in set (0.00 sec)

| sys           | ps_setup_reload_saved               | PROCEDURE | mysql.sys@localhost | 2020-06-0
Description

Reloads a saved Performance Schema configuration,
so that you can alter the setup for debugging purposes,
but restore it to a previous state.

Use the companion procedure - ps_setup_save(), to
save a configuration.

Requires the SUPER privilege for "SET sql_log_bin = 0;".

Parameters

None.

Example

mysql> CALL sys.ps_setup_save();
Query OK, 0 rows affected (0.08 sec)

mysql> UPDATE performance_schema.setup_instruments SET enabled = 'YES', timed = 'YES';
Query OK, 547 rows affected (0.40 sec)
Rows matched: 784  Changed: 547  Warnings: 0

/* Run some tests that need more detailed instrumentation here */

mysql> CALL sys.ps_setup_reload_saved();
Query OK, 0 rows affected (0.32 sec)

| sys           | ps_setup_reset_to_default           | PROCEDURE | mysql.sys@localhost | 2020-06-0
Description

Resets the Performance Schema setup to the default settings.

Parameters

in_verbose (BOOLEAN):
Whether to print each setup stage (including the SQL) whilst running.

Example

mysql> CALL sys.ps_setup_reset_to_default(true)\G
*************************** 1. row ***************************
status: Resetting: setup_actors
DELETE
FROM performance_schema.setup_actors
WHERE NOT (HOST = '%' AND USER = '%' AND `ROLE` = '%')
1 row in set (0.00 sec)

*************************** 1. row ***************************
status: Resetting: setup_actors
INSERT IGNORE INTO performance_schema.setup_actors
VALUES ('%', '%', '%')
1 row in set (0.00 sec)
...

mysql> CALL sys.ps_setup_reset_to_default(false)\G
```

```
Query OK, 0 rows affected (0.00 sec)

| sys           | ps_setup_save                       | PROCEDURE | mysql.sys@localhost | 2020-06-0
Description

Saves the current configuration of Performance Schema,
so that you can alter the setup for debugging purposes,
but restore it to a previous state.

Use the companion procedure - ps_setup_reload_saved(), to
restore the saved config.

The named lock "sys.ps_setup_save" is taken before the
current configuration is saved. If the attempt to get the named
lock times out, an error occurs.

The lock is released after the settings have been restored by
calling ps_setup_reload_saved().

Requires the SUPER privilege for "SET sql_log_bin = 0;".

Parameters

in_timeout INT
The timeout in seconds used when trying to obtain the lock.
A negative timeout means infinite timeout.

Example

mysql> CALL sys.ps_setup_save(-1);
Query OK, 0 rows affected (0.08 sec)

mysql> UPDATE performance_schema.setup_instruments
->    SET enabled = 'YES', timed = 'YES';
Query OK, 547 rows affected (0.40 sec)
Rows matched: 784  Changed: 547  Warnings: 0

/* Run some tests that need more detailed instrumentation here */

mysql> CALL sys.ps_setup_reload_saved();
Query OK, 0 rows affected (0.32 sec)
  | utf8                 | utf8_general_ci      | utf8_general_ci    |
| sys           | ps_setup_show_disabled              | PROCEDURE | mysql.sys@localhost | 2020-06-0
Description

Shows all currently disable Performance Schema configuration.

Disabled users is only available for MySQL 5.7.6 and later.
In earlier versions it was only possible to enable users.

Parameters

in_show_instruments (BOOLEAN):
Whether to print disabled instruments (can print many items)

in_show_threads (BOOLEAN):
Whether to print disabled threads

Example

mysql> CALL sys.ps_setup_show_disabled(TRUE, TRUE);
+----------------------------+
```

```
| performance_schema_enabled |
+----------------------------+
|                          1 |
+----------------------------+
1 row in set (0.00 sec)

+--------------------+
| disabled_users     |
+--------------------+
| 'mark'@'localhost' |
+--------------------+
1 row in set (0.00 sec)

+-------------+----------------------+---------+-------+
| object_type | objects              | enabled | timed |
+-------------+----------------------+---------+-------+
| EVENT       | mysql.%              | NO      | NO    |
| EVENT       | performance_schema.% | NO      | NO    |
| EVENT       | information_schema.% | NO      | NO    |
| FUNCTION    | mysql.%              | NO      | NO    |
| FUNCTION    | performance_schema.% | NO      | NO    |
| FUNCTION    | information_schema.% | NO      | NO    |
| PROCEDURE   | mysql.%              | NO      | NO    |
| PROCEDURE   | performance_schema.% | NO      | NO    |
| PROCEDURE   | information_schema.% | NO      | NO    |
| TABLE       | mysql.%              | NO      | NO    |
| TABLE       | performance_schema.% | NO      | NO    |
| TABLE       | information_schema.% | NO      | NO    |
| TRIGGER     | mysql.%              | NO      | NO    |
| TRIGGER     | performance_schema.% | NO      | NO    |
| TRIGGER     | information_schema.% | NO      | NO    |
+-------------+----------------------+---------+-------+
15 rows in set (0.00 sec)

+---------------------------------+
| disabled_consumers              |
+---------------------------------+
| events_stages_current           |
| events_stages_history           |
| events_stages_history_long      |
| events_statements_history       |
| events_statements_history_long  |
| events_transactions_history     |
| events_transactions_history_long |
| events_waits_current            |
| events_waits_history            |
| events_waits_history_long       |
+---------------------------------+
10 rows in set (0.00 sec)

Empty set (0.00 sec)

+-------------------------------------------------------------------------------------------+-------+
| disabled_instruments                                                                      | timed |
+-------------------------------------------------------------------------------------------+-------+
| wait/synch/mutex/sql/TC_LOG_MMAP::LOCK_tc                                                 | NO    |
| wait/synch/mutex/sql/LOCK_des_key_file                                                    | NO    |
| wait/synch/mutex/sql/MYSQL_BIN_LOG::LOCK_commit                                           | NO    |
...
| memory/sql/servers_cache                                                                  | NO    |
| memory/sql/udf_mem                                                                        | NO    |
| wait/lock/metadata/sql/mdl                                                                | NO    |
```

```
+-------------------------------------------------------------------------------------+-------+
547 rows in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
  | utf8                 | utf8_general_ci      | utf8_general_ci    |
| sys           | ps_setup_show_disabled_consumers    | PROCEDURE | mysql.sys@localhost | 2020-06-0
Description

Shows all currently disabled consumers.

Parameters

None

Example

mysql> CALL sys.ps_setup_show_disabled_consumers();

+---------------------------+
| disabled_consumers        |
+---------------------------+
| events_statements_current |
| global_instrumentation    |
| thread_instrumentation    |
| statements_digest         |
+---------------------------+
4 rows in set (0.05 sec)

| sys           | ps_setup_show_disabled_instruments  | PROCEDURE | mysql.sys@localhost | 2020-06-0
Description

Shows all currently disabled instruments.

Parameters

None

Example

mysql> CALL sys.ps_setup_show_disabled_instruments();

| sys           | ps_setup_show_enabled               | PROCEDURE | mysql.sys@localhost | 2020-06-0
Description

Shows all currently enabled Performance Schema configuration.

Parameters

in_show_instruments (BOOLEAN):
Whether to print enabled instruments (can print many items)

in_show_threads (BOOLEAN):
Whether to print enabled threads

Example

mysql> CALL sys.ps_setup_show_enabled(TRUE, TRUE);
+---------------------------+
| performance_schema_enabled |
+---------------------------+
|                         1 |
+---------------------------+
```

```
1 row in set (0.00 sec)

+---------------+
| enabled_users |
+---------------+
| '%'@'%'       |
+---------------+
1 row in set (0.01 sec)

+-------------+---------+---------+-------+
| object_type | objects | enabled | timed |
+-------------+---------+---------+-------+
| EVENT       | %.%     | YES     | YES   |
| FUNCTION    | %.%     | YES     | YES   |
| PROCEDURE   | %.%     | YES     | YES   |
| TABLE       | %.%     | YES     | YES   |
| TRIGGER     | %.%     | YES     | YES   |
+-------------+---------+---------+-------+
5 rows in set (0.01 sec)

+---------------------------+
| enabled_consumers         |
+---------------------------+
| events_statements_current |
| global_instrumentation    |
| thread_instrumentation    |
| statements_digest         |
+---------------------------+
4 rows in set (0.05 sec)

+---------------------------------+-------------+
| enabled_threads                 | thread_type |
+---------------------------------+-------------+
| sql/main                        | BACKGROUND  |
| sql/thread_timer_notifier       | BACKGROUND  |
| innodb/io_ibuf_thread           | BACKGROUND  |
| innodb/io_log_thread            | BACKGROUND  |
| innodb/io_read_thread           | BACKGROUND  |
| innodb/io_read_thread           | BACKGROUND  |
| innodb/io_write_thread          | BACKGROUND  |
| innodb/io_write_thread          | BACKGROUND  |
| innodb/page_cleaner_thread      | BACKGROUND  |
| innodb/srv_lock_timeout_thread  | BACKGROUND  |
| innodb/srv_error_monitor_thread | BACKGROUND  |
| innodb/srv_monitor_thread       | BACKGROUND  |
| innodb/srv_master_thread        | BACKGROUND  |
| innodb/srv_purge_thread         | BACKGROUND  |
| innodb/srv_worker_thread        | BACKGROUND  |
| innodb/srv_worker_thread        | BACKGROUND  |
| innodb/srv_worker_thread        | BACKGROUND  |
| innodb/buf_dump_thread          | BACKGROUND  |
| innodb/dict_stats_thread        | BACKGROUND  |
| sql/signal_handler              | BACKGROUND  |
| sql/compress_gtid_table         | FOREGROUND  |
| root@localhost                  | FOREGROUND  |
+---------------------------------+-------------+
22 rows in set (0.01 sec)

+-----------------------------------+-------+
| enabled_instruments               | timed |
+-----------------------------------+-------+
| wait/io/file/sql/map              | YES   |
```

```
| wait/io/file/sql/binlog             | YES   |
...
| statement/com/Error                 | YES   |
| statement/com/                      | YES   |
| idle                                | YES   |
+-------------------------------------+-------+
210 rows in set (0.08 sec)

Query OK, 0 rows affected (0.89 sec)
 | utf8                 | utf8_general_ci      | utf8_general_ci    |
| sys           | ps_setup_show_enabled_consumers     | PROCEDURE | mysql.sys@localhost | 2020-06-
Description

Shows all currently enabled consumers.

Parameters

None

Example

mysql> CALL sys.ps_setup_show_enabled_consumers();

+---------------------------+
| enabled_consumers         |
+---------------------------+
| events_statements_current |
| global_instrumentation    |
| thread_instrumentation    |
| statements_digest         |
+---------------------------+
4 rows in set (0.05 sec)

| sys           | ps_setup_show_enabled_instruments   | PROCEDURE | mysql.sys@localhost | 2020-06-
Description

Shows all currently enabled instruments.

Parameters

None

Example

mysql> CALL sys.ps_setup_show_enabled_instruments();

| sys           | ps_statement_avg_latency_histogram  | PROCEDURE | mysql.sys@localhost | 2020-06-
Description

Outputs a textual histogram graph of the average latency values
across all normalized queries tracked within the Performance Schema
events_statements_summary_by_digest table.

Can be used to show a very high level picture of what kind of
latency distribution statements running within this instance have.

Parameters

None.

Example
```

```
mysql> CALL sys.ps_statement_avg_latency_histogram()\G
*************************** 1. row ***************************
Performance Schema Statement Digest Average Latency Histogram:

. = 1 unit
* = 2 units
# = 3 units

(0 - 38ms)     240 | ########################################################################################
(38 - 77ms)    38  | ......................................
(77 - 115ms)   3   | ...
(115 - 154ms)  62  | ******************************
(154 - 192ms)  3   | ...
(192 - 231ms)  0   |
(231 - 269ms)  0   |
(269 - 307ms)  0   |
(307 - 346ms)  0   |
(346 - 384ms)  1   | .
(384 - 423ms)  1   | .
(423 - 461ms)  0   |
(461 - 499ms)  0   |
(499 - 538ms)  0   |
(538 - 576ms)  0   |
(576 - 615ms)  1   | .

Total Statements: 350; Buckets: 16; Bucket Size: 38 ms;
  | utf8                  | utf8_general_ci      | utf8_general_ci    |
| sys           | ps_trace_statement_digest           | PROCEDURE | mysql.sys@localhost | 2020-06-0
Description

Traces all instrumentation within Performance Schema for a specific
Statement Digest.

When finding a statement of interest within the
performance_schema.events_statements_summary_by_digest table, feed
the DIGEST MD5 value in to this procedure, set how long to poll for,
and at what interval to poll, and it will generate a report of all
statistics tracked within Performance Schema for that digest for the
interval.

It will also attempt to generate an EXPLAIN for the longest running
example of the digest during the interval. Note this may fail, as:

* Performance Schema truncates long SQL_TEXT values (and hence the
EXPLAIN will fail due to parse errors)
* the default schema is sys (so tables that are not fully qualified
in the query may not be found)
* some queries such as SHOW are not supported in EXPLAIN.

When the EXPLAIN fails, the error will be ignored and no EXPLAIN
output generated.

Requires the SUPER privilege for "SET sql_log_bin = 0;".

Parameters

in_digest (VARCHAR(32)):
The statement digest identifier you would like to analyze
in_runtime (INT):
The number of seconds to run analysis for
in_interval (DECIMAL(2,2)):
The interval (in seconds, may be fractional) at which to try
```

```
and take snapshots
in_start_fresh (BOOLEAN):
Whether to TRUNCATE the events_statements_history_long and
events_stages_history_long tables before starting
in_auto_enable (BOOLEAN):
Whether to automatically turn on required consumers

Example

mysql> call ps_trace_statement_digest('891ec6860f98ba46d89dd20b0c03652c', 10, 0.1, true, true);
+--------------------+
| SUMMARY STATISTICS |
+--------------------+
| SUMMARY STATISTICS |
+--------------------+
1 row in set (9.11 sec)

+------------+-----------+-----------+-----------+---------------+------------+------------+
| executions | exec_time | lock_time | rows_sent | rows_examined | tmp_tables | full_scans |
+------------+-----------+-----------+-----------+---------------+------------+------------+
|         21 | 4.11 ms   | 2.00 ms   |         0 |            21 |          0 |          0 |
+------------+-----------+-----------+-----------+---------------+------------+------------+
1 row in set (9.11 sec)

+----------------------------------------+-------+-----------+
| event_name                             | count | latency   |
+----------------------------------------+-------+-----------+
| stage/sql/checking query cache for query |    16 | 724.37 us |
| stage/sql/statistics                   |    16 | 546.92 us |
| stage/sql/freeing items                |    18 | 520.11 us |
| stage/sql/init                         |    51 | 466.80 us |
...
| stage/sql/cleaning up                  |    18 | 11.92 us  |
| stage/sql/executing                    |    16 | 6.95 us   |
+----------------------------------------+-------+-----------+
17 rows in set (9.12 sec)

+---------------------------+
| LONGEST RUNNING STATEMENT |
+---------------------------+
| LONGEST RUNNING STATEMENT |
+---------------------------+
1 row in set (9.16 sec)

+-----------+-----------+-----------+-----------+---------------+------------+-----------+
| thread_id | exec_time | lock_time | rows_sent | rows_examined | tmp_tables | full_scan |
+-----------+-----------+-----------+-----------+---------------+------------+-----------+
|    166646 | 618.43 us | 1.00 ms   |         0 |             1 |          0 |         0 |
+-----------+-----------+-----------+-----------+---------------+------------+-----------+
1 row in set (9.16 sec)

// Truncated for clarity...
+-------------------------------------------------------------------+
| sql_text                                                          |
+-------------------------------------------------------------------+
| select hibeventhe0_.id as id1382_, hibeventhe0_.createdTime ... |
+-------------------------------------------------------------------+
1 row in set (9.17 sec)

+----------------------------------------+-----------+
| event_name                             | latency   |
+----------------------------------------+-----------+
```

```
| stage/sql/init                          | 8.61 us   |
| stage/sql/Waiting for query cache lock  | 453.23 us |
| stage/sql/init                          | 331.07 ns |
| stage/sql/checking query cache for query | 43.04 us  |
...
| stage/sql/freeing items                 | 30.46 us  |
| stage/sql/cleaning up                   | 662.13 ns |
+-----------------------------------------+-----------+
18 rows in set (9.23 sec)

+----+-------------+-------------+-------+---------------+-----------+---------+-------------+--
| id | select_type | table       | type  | possible_keys | key       | key_len | ref         | ro
+----+-------------+-------------+-------+---------------+-----------+---------+-------------+--
|  1 | SIMPLE      | hibeventhe0_ | const | fixedTime     | fixedTime | 775     | const,const |
+----+-------------+-------------+-------+---------------+-----------+---------+-------------+--
1 row in set (9.27 sec)

Query OK, 0 rows affected (9.28 sec)
 | utf8                | utf8_general_ci      | utf8_general_ci    |
| sys           | ps_trace_thread                     | PROCEDURE | mysql.sys@localhost | 2020-06-0
Description

Dumps all data within Performance Schema for an instrumented thread,
to create a DOT formatted graph file.

Each resultset returned from the procedure should be used for a complete graph

Requires the SUPER privilege for "SET sql_log_bin = 0;".

Parameters

in_thread_id (BIGINT UNSIGNED):
The thread that you would like a stack trace for
in_outfile  (VARCHAR(255)):
The filename the dot file will be written to
in_max_runtime (DECIMAL(20,2)):
The maximum time to keep collecting data.
Use NULL to get the default which is 60 seconds.
in_interval (DECIMAL(20,2)):
How long to sleep between data collections.
Use NULL to get the default which is 1 second.
in_start_fresh (BOOLEAN):
Whether to reset all Performance Schema data before tracing.
in_auto_setup (BOOLEAN):
Whether to disable all other threads and enable all consumers/instruments.
This will also reset the settings at the end of the run.
in_debug (BOOLEAN):
Whether you would like to include file:lineno in the graph

Example

mysql> CALL sys.ps_trace_thread(25, CONCAT('/tmp/stack-', REPLACE(NOW(), ' ', '-'), '.dot'), NULL
+--------------------+
| summary            |
+--------------------+
| Disabled 1 thread |
+--------------------+
1 row in set (0.00 sec)

+-------------------------------------------+
| Info                                      |
+-------------------------------------------+
```

```
| Data collection starting for THREAD_ID = 25 |
+---------------------------------------------+
1 row in set (0.03 sec)

+-----------------------------------------------------------+
| Info                                                      |
+-----------------------------------------------------------+
| Stack trace written to /tmp/stack-2014-02-16-21:18:41.dot |
+-----------------------------------------------------------+
1 row in set (60.07 sec)

+-------------------------------------------------------------------+
| Convert to PDF                                                    |
+-------------------------------------------------------------------+
| dot -Tpdf -o /tmp/stack_25.pdf /tmp/stack-2014-02-16-21:18:41.dot |
+-------------------------------------------------------------------+
1 row in set (60.07 sec)

+-------------------------------------------------------------------+
| Convert to PNG                                                    |
+-------------------------------------------------------------------+
| dot -Tpng -o /tmp/stack_25.png /tmp/stack-2014-02-16-21:18:41.dot |
+-------------------------------------------------------------------+
1 row in set (60.07 sec)

+------------------+
| summary          |
+------------------+
| Enabled 1 thread |
+------------------+
1 row in set (60.32 sec)
  | utf8                 | utf8_general_ci      | utf8_general_ci    |
| sys           | ps_truncate_all_tables              | PROCEDURE | mysql.sys@localhost | 2020-06-
 Description

 Truncates all summary tables within Performance Schema,
 resetting all aggregated instrumentation as a snapshot.

 Parameters

 in_verbose (BOOLEAN):
 Whether to print each TRUNCATE statement before running

 Example

 mysql> CALL sys.ps_truncate_all_tables(false);
 +---------------------+
 | summary             |
 +---------------------+
 | Truncated 44 tables |
 +---------------------+
 1 row in set (0.10 sec)

 Query OK, 0 rows affected (0.10 sec)

| sys           | statement_performance_analyzer      | PROCEDURE | mysql.sys@localhost | 2020-06-
 Description

 Create a report of the statements running on the server.

 The views are calculated based on the overall and/or delta activity.
```

```
Requires the SUPER privilege for "SET sql_log_bin = 0;".

Parameters

in_action (ENUM('snapshot', 'overall', 'delta', 'create_tmp', 'create_table', 'save', 'cleanup'))
The action to take. Supported actions are:
* snapshot      Store a snapshot. The default is to make a snapshot of the current content of
performance_schema.events_statements_summary_by_digest, but by setting in_table
this can be overwritten to copy the content of the specified table.
The snapshot is stored in the sys.tmp_digests temporary table.
* overall       Generate analyzis based on the content specified by in_table. For the overall ana
in_table can be NOW() to use a fresh snapshot. This will overwrite an existing snapshot.
Use NULL for in_table to use the existing snapshot. If in_table IS NULL and no snapshot
exists, a new will be created.
See also in_views and @sys.statement_performance_analyzer.limit.
* delta         Generate a delta analysis. The delta will be calculated between the reference tab
in_table and the snapshot. An existing snapshot must exist.
The action uses the sys.tmp_digests_delta temporary table.
See also in_views and @sys.statement_performance_analyzer.limit.
* create_table  Create a regular table suitable for storing the snapshot for later use, e.g. for
calculating deltas.
* create_tmp    Create a temporary table suitable for storing the snapshot for later use, e.g. fo
calculating deltas.
* save          Save the snapshot in the table specified by in_table. The table must exists and ha
the correct structure.
If no snapshot exists, a new is created.
* cleanup       Remove the temporary tables used for the snapshot and delta.

in_table (VARCHAR(129)):
The table argument used for some actions. Use the format 'db1.t1' or 't1' without using any backt
for quoting. Periods (.) are not supported in the database and table names.

The meaning of the table for each action supporting the argument is:

* snapshot      The snapshot is created based on the specified table. Set to NULL or NOW() to use
the current content of performance_schema.events_statements_summary_by_digest.
* overall       The table with the content to create the overall analyzis for. The following valu
can be used:
- A table name - use the content of that table.
- NOW()        - create a fresh snapshot and overwrite the existing snapshot.
- NULL         - use the last stored snapshot.
* delta         The table name is mandatory and specified the reference view to compare the curre
stored snapshot against. If no snapshot exists, a new will be created.
* create_table  The name of the regular table to create.
* create_tmp    The name of the temporary table to create.
* save          The name of the table to save the currently stored snapshot into.

in_views (SET ('with_runtimes_in_95th_percentile', 'analysis', 'with_errors_or_warnings',
'with_full_table_scans', 'with_sorting', 'with_temp_tables', 'custom'))
Which views to include:  * with_runtimes_in_95th_percentile  Based on the sys.statements_with_runt
| sys           | table_exists                        | PROCEDURE | mysql.sys@localhost | 2020-06-0
Description

Tests whether the table specified in in_db and in_table exists either as a regular
table, or as a temporary table. The returned value corresponds to the table that
will be used, so if there's both a temporary and a permanent table with the given
name, then 'TEMPORARY' will be returned.

Parameters

in_db (VARCHAR(64)):
The database name to check for the existance of the table in.
```

```
in_table (VARCHAR(64)):
The name of the table to check the existance of.

out_exists ENUM('', 'BASE TABLE', 'VIEW', 'TEMPORARY'):
The return value: whether the table exists. The value is one of:
* ''           - the table does not exist neither as a base table, view, nor temporary table.
* 'BASE TABLE' - the table name exists as a permanent base table table.
* 'VIEW'       - the table name exists as a view.
* 'TEMPORARY'  - the table name exists as a temporary table.

Example

mysql> CREATE DATABASE db1;
Query OK, 1 row affected (0.07 sec)

mysql> use db1;
Database changed
mysql> CREATE TABLE t1 (id INT PRIMARY KEY);
Query OK, 0 rows affected (0.08 sec)

mysql> CREATE TABLE t2 (id INT PRIMARY KEY);
Query OK, 0 rows affected (0.08 sec)

mysql> CREATE view v_t1 AS SELECT * FROM t1;
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE TEMPORARY TABLE t1 (id INT PRIMARY KEY);
Query OK, 0 rows affected (0.00 sec)

mysql> CALL sys.table_exists('db1', 't1', @exists); SELECT @exists;
Query OK, 0 rows affected (0.00 sec)

+------------+
| @exists    |
+------------+
| TEMPORARY  |
+------------+
1 row in set (0.00 sec)

mysql> CALL sys.table_exists('db1', 't2', @exists); SELECT @exists;
Query OK, 0 rows affected (0.00 sec)

+------------+
| @exists    |
+------------+
| BASE TABLE |
+------------+
1 row in set (0.01 sec)

mysql> CALL sys.table_exists('db1', 'v_t1', @exists); SELECT @exists;
Query OK, 0 rows affected (0.00 sec)

+---------+
| @exists |
+---------+
| VIEW    |
+---------+
1 row in set (0.00 sec)

mysql> CALL sys.table_exists('db1', 't3', @exists); SELECT @exists;
Query OK, 0 rows affected (0.01 sec)
```

```
 +---------+
 | @exists |
 +---------+
 |         |
 +---------+
 1 row in set (0.00 sec)
  | utf8                 | utf8_general_ci      | utf8_general_ci    |
+--------------+------------------------------------+-----------+--------------------+---------
27 rows in set (0.00 sec)
```

3. A **WHERE** clause can be used with the above statement to view the stored procedures of any database by specifying the name of the database. For example:

```
SHOW PROCEDURE STATUS WHERE db = 'MovieIndustry';
```

The **information_schema** database contains the data about all the stored procedures in all the databases. We can query the **routines** table in this database as follows:

```
SELECT routine_name
FROM information_schema.routines
WHERE routine_type = 'PROCEDURE'
  AND routine_schema = 'sys';
```

the above query lists 26 procedures in the **sys** database.

4. To delete the stored procedure we just created execute the following query:

```
DROP PROCEDURE IF EXISTS ShowActors;
```

The **IF EXISTS** clause in this statement is optional. It is used to avoid an error message in case we attempt to delete a stored procedure that does not exist. A warning is issued instead.

As it can be seen, **ShowActors()** has been successfully deleted from our database.

5. It is possible to make changes to a stored procedure. However, there are no MySQL statements that can directly modify the parameter list or the body of the stored procedure. To make changes to a stored procedure, the only way is to delete the procedure and then re-create it.