

```
SELECT * FROM performance_schema.mutex_instances
WHERE OBJECT_INSTANCE_BEGIN = mutex_A;
```

Say the query result identifies that it is thread 2 holding mutex A, as found in `mutex_instances.LOCKED_BY_THREAD_ID`.

4. You can see what thread 2 is doing:

```
SELECT * FROM performance_schema.events_waits_current
WHERE THREAD_ID = thread_2;
```

27.19.1 Query Profiling Using Performance Schema

The following example demonstrates how to use Performance Schema statement events and stage events to retrieve data comparable to profiling information provided by `SHOW PROFILES` and `SHOW PROFILE` statements.

The `setup_actors` table can be used to limit the collection of historical events by host, user, or account to reduce runtime overhead and the amount of data collected in history tables. The first step of the example shows how to limit collection of historical events to a specific user.

Performance Schema displays event timer information in picoseconds (trillionths of a second) to normalize timing data to a standard unit. In the following example, `TIMER_WAIT` values are divided by 1000000000000 to show data in units of seconds. Values are also truncated to 6 decimal places to display data in the same format as `SHOW PROFILES` and `SHOW PROFILE` statements.

1. Limit the collection of historical events to the user that runs the query. By default, `setup_actors` is configured to allow monitoring and historical event collection for all foreground threads:

```
mysql> SELECT * FROM performance_schema.setup_actors;
+-----+-----+-----+-----+-----+
| HOST | USER | ROLE | ENABLED | HISTORY |
+-----+-----+-----+-----+-----+
| %    | %    | %    | YES     | YES     |
+-----+-----+-----+-----+-----+
```

Update the default row in the `setup_actors` table to disable historical event collection and monitoring for all foreground threads, and insert a new row that enables monitoring and historical event collection for the user that runs the query:

```
mysql> UPDATE performance_schema.setup_actors
      SET ENABLED = 'NO', HISTORY = 'NO'
      WHERE HOST = '%' AND USER = '%';

mysql> INSERT INTO performance_schema.setup_actors
      (HOST,USER,ROLE,ENABLED,HISTORY)
      VALUES('localhost','test_user','%','YES','YES');
```

Data in the `setup_actors` table should now appear similar to the following:

```
mysql> SELECT * FROM performance_schema.setup_actors;
+-----+-----+-----+-----+-----+
| HOST      | USER      | ROLE | ENABLED | HISTORY |
+-----+-----+-----+-----+-----+
| %         | %         | %    | NO      | NO      |
| localhost | test_user | %    | YES     | YES     |
+-----+-----+-----+-----+-----+
```

2. Ensure that statement and stage instrumentation is enabled by updating the `setup_instruments` table. Some instruments may already be enabled by default.