Maulik Bhatt

MySQL Query Log Analysis Using pt-query-digest

Contents

[MySQL Query Log Analysis Using pt-query-digest 2](#_Toc194072380)

[Task Description 2](#_Toc194072381)

[Query Log Analysis Output 2](#_Toc194072382)

[Overall Statistics 2](#_Toc194072383)

[Query Performance Profile 2](#_Toc194072384)

[Query 1: INSERT orders\_30 2](#_Toc194072385)

[Query 2: SELECT orders\_67 3](#_Toc194072386)

[Query 3: INSERT orders\_77 3](#_Toc194072387)

[Additional Observations 3](#_Toc194072388)

[Customization Options for pt-query-digest 4](#_Toc194072389)

[1. Adjusting the Number of Queries in the Report 4](#_Toc194072390)

[2. Filtering by Query Execution Time 4](#_Toc194072391)

[3. Filtering by Database 4](#_Toc194072392)

[4. Filtering by Query Type 4](#_Toc194072393)

[5. Saving the Report to a File 4](#_Toc194072394)

[Conclusion 4](#_Toc194072395)

# MySQL Query Log Analysis Using pt-query-digest

## Task Description

I have restored a dump of test\_db\_1 for testing log analysis using pt-query-digest. The purpose of this analysis is to identify slow queries and performance bottlenecks in MySQL.

## Query Log Analysis Output

### Overall Statistics

* **Total Queries Analyzed:** 530
* **Unique Queries:** 5
* **Query Execution Rate:** 0.20 QPS
* **Query Concurrency:** 0.26x
* **Total Execution Time:** 665 seconds
* **Rows Sent:** 12.18M
* **Rows Examined:** 12.18M
* **Total Query Size:** 425.73MB

### Query Performance Profile

| Rank | Query ID | Response Time (s) | Calls | Avg. Exec Time (s) | Type |
| --- | --- | --- | --- | --- | --- |
| 1 | 0x709962148C060B142F18D9E28404257B | 416.417 (62.6%) | 361 | 1.1535 | INSERT orders\_? |
| 2 | 0xE3C753C2F267B2D767A347A2812914DF | 148.623 (22.3%) | 99 | 1.5012 | SELECT orders\_? |
| 3 | 0x4C40C2EADB941ED3044863B266EE721F | 79.257 (11.9%) | 68 | 1.1655 | INSERT orders\_? |

### Query 1: INSERT orders\_30

* **Query ID:** 0x709962148C060B142F18D9E28404257B
* **Execution Time:** 416s (62% of total execution time)
* **Calls:** 361
* **Average Execution Time:** 1.1535s
* **Query Example:**
* INSERT INTO `orders\_30` VALUES (63129, '402-0528390-8964303', '05-20-22', \_binary 'Shipped - Delivered to Buyer', 'Merchant', 'Amazon.in', 'Standard', 'SET341', \_binary 'SET341-KR-NP-S', 'Set', 'S', 'B09NPWMVFT', 'Shipped', 1, 'INR', 857.00, \_binary 'Pune', \_binary 'MAHARASHTRA', '411040', 'IN', \_binary 'Amazon PLCC Free-Financing Universal Merchant AAT-WNKTBO3K27EJC,Amazon PLCC Free-Financing Universal Merchant AAT-QX3UCCJESKPA2,Amazon PLCC Free-Financing Universal Merchant AAT-5QQ7BIYYQEDN2','FALSE', 'Easy Ship');

### Query 2: SELECT orders\_67

* **Query ID:** 0xE3C753C2F267B2D767A347A2812914DF
* **Execution Time:** 148s (22% of total execution time)
* **Calls:** 99
* **Average Execution Time:** 1.5012s
* **Rows Sent:** 12.18M
* **Rows Examined:** 12.18M
* **Query Example:**
* SELECT \* FROM `orders\_67`;

### Query 3: INSERT orders\_77

* **Query ID:** 0x4C40C2EADB941ED3044863B266EE721F
* **Execution Time:** 79s (11.9% of total execution time)
* **Calls:** 68
* **Average Execution Time:** 1.1655s
* **Query Example:**
* INSERT INTO `orders\_77` VALUES (0, '405-8078784-5731545', '04-30-22', \_binary 'Cancelled', 'Merchant', 'Amazon.in', 'Standard', 'SET389', \_binary 'SET389-KR-NP-S', 'Set', 'S', 'B09KXVBD7Z', '', 0, 'INR', 647.62, \_binary 'MUMBAI', \_binary 'MAHARASHTRA', '400081', 'IN', '', 'FALSE', 'Easy Ship');

## Additional Observations

* pt-query-digest only provides details for the top 3 most time-consuming queries by default.
* Queries involving INSERT INTO orders\_\* consume most of the execution time.
* The SELECT query on orders\_67 retrieves a large number of rows (12.18M), indicating possible inefficiencies.

## Customization Options for pt-query-digest

### 1. Adjusting the Number of Queries in the Report

By default, pt-query-digest shows the top 3 most time-consuming queries. This can be modified using --limit:

pt-query-digest --limit=5 /var/log/mysql/srvtraining-ubuntu-slow.log

This will display the top 5 queries instead of 3.

### 2. Filtering by Query Execution Time

To analyze queries that took longer than a certain time:

pt-query-digest --filter 'Query\_time > 2' /var/log/mysql/srvtraining-ubuntu-slow.log

This filters out queries that executed in less than 2 seconds.

### 3. Filtering by Database

To analyze queries from a specific database:

pt-query-digest --filter 'db == "test\_db\_1"' /var/log/mysql/srvtraining-ubuntu-slow.log

### 4. Filtering by Query Type

To focus only on SELECT queries:

pt-query-digest --filter 'fingerprint =~ "^SELECT"' /var/log/mysql/srvtraining-ubuntu-slow.log

### 5. Saving the Report to a File

To save the analysis report instead of displaying it in the terminal:

pt-query-digest /var/log/mysql/srvtraining-ubuntu-slow.log > report.txt

This allows for easier review and sharing.

## Conclusion

The pt-query-digest tool helps in identifying query bottlenecks, and its customization options allow deeper filtering and reporting based on specific needs.