# Learn how to Analyze & Tune MySQL Queries for Better Performance

## Tools and techniques for faster queries

**Bradley Mickel**

MySQL DBA

Thursday June 21st

PERCONA

# Tools

Overview of common tools used to identify query optimizations

# EXPLAIN

EXPLAIN SELECT * FROM staff;

EXPLAIN SELECT * FROM staff\G

| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | SIMPLE | staff | NULL | ALL | NULL | NULL | NULL | NULL | 2 | 100.00 | NULL |

```
           id: 1
  select_type: SIMPLE
        table: staff
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 2
     filtered: 100.00
        Extra: NULL
1 row in set, 1 warning (0.00 sec)
```





PERCONA

# JSON FORMAT EXPLAIN

EXPLAIN FORMAT=JSON SELECT count(1) FROM sakila.rental WHERE last_update > ‘2006-02-18 00:00:00'\G

```
EXPLAIN: {
  "query_block": {
    "select_id": 1,
    "cost_info": {
      "query_cost": "1.16"
    },
    "table": {
      "table_name": "rental",
      "access_type": "range",
      "possible_keys": [
        "last_update"
      ],
      "key": "last_update",
      "used_key_parts": [
        "last_update"
      ],
      "key_length": "4",
      "rows_examined_per_scan": 1,
      "rows_produced_per_join": 1,
      "filtered": "100.00",
      "using_index": true,
      "cost_info": {
        "read_cost": "1.06",
        "eval_cost": "0.10",
        "prefix_cost": "1.16",
        "data_read_per_join": "32"
      },
      "used_columns": [
        "last_update"
      ],
      "attached_condition": "(`sakila`.`rental`.`last_update` > '2006-02-18 00:00:00')"
    }
  }
}
```





PERCONA

# EXPLAIN Fields

| Field | JSON FIELD | Description |
|---|---|---|
| id | select_id | The sequential number of the SELECT within the query |
| select_type | | The Type of select being performed. |
| table | table_name | The name, or alias of the table related to this step of the queries |
| partitions | partitions | The partitions from which records would be matched by the query. NULL for non partitioned tables |
| type | access_type | The type of access being used to retrieve data or join tables |
| possible_keys | possible_keys | Which indexes are being reviewed for use in the query |
| key | key | The index which will be used for the query |
| key_len | key_length | The length of the key that MySQL has decided to use. This enables you to determine how many parts of a composite index are being used.* |
| ref | ref | Shows which columns or constants are compared to the index named in the key column to select rows from the table |
| rows | rows | The number of rows MySQL estimates will be need to be reviewed |
| filtered | filtered | An estimated percentage of table rows that will be filtered by the condition. 100% means all the rows returned by the condition will be joined with previous tables |
| extra | | Additional information related to the optimizer's chosen path. |

# JSON FORMAT EXPLAIN

```
EXPLAIN: {
  "query_block": {
    "select_id": 1,
    "cost_info": {
      "query_cost": "0.62"
    },
    "table": {
      "table_name": "rental",
      "access_type": "range",
      "possible_keys": [
        "last_update"
      ],
      "key": "last_update",
      "used_key_parts": [
        "last_update"
      ],
      "key_length": "4",
```

- Additional Fields:
  - *cost_Info*
    - *query_cost*
  - *used_key_parts*





PERCONA

# JSON FORMAT EXPLAIN

```
      "rows_examined_per_scan": 1,
      "rows_produced_per_join": 1,
      "filtered": "100.00",
      "using_index": true,
      "cost_info": {
        "read_cost": "0.52",
        "eval_cost": "0.10",
        "prefix_cost": "0.62",
        "data_read_per_join": "32"
      },
      "used_columns": [
        "last_update"
      ],
      "attached_condition": "(`sakila`.`rent
al`.`last_update` > '2006-02-18 00:00:00')"
    }
  }
}
```

- Additional Fields:
  - *rows_examined_per_scan*
  - *rows_examined_per_join*
  - *cost_info*
    - *read_cost*
    - *eval_cost*
    - *prefix_cost*
    - *data_read_per_join*
  - *used_columns*

# Access Types

| Type | Description |
|---|---|
| constant | The table has at most one matching row which is read at the start of the query |
| system | A type of constant table used for systems tables |
| eq_ref | Used for table joins with a 1:1 relationship. Requires the joining index by a Primary or Unique Key |
| ref | All rows with matching index values are read from this table for each combination of rows from the previous tables |
| fulltext | A fulltext Index is being used |
| ref_or_null | Similar to ref, but will find null values in addition to the selected value |
| index_merge | Index Merge is being used for the join |
| unique_subquery | Replaces eq_ref for certain subqueries using IN |
| index_subquery | Similar to unique_subquery but for when the index does not have a unique constraint |
| range | Only rows that are in a given range are retrieved, using an index to select the rows. The key column in the output row indicates which index is used. The key_len contains the longest key part that was used. The ref column is NULL for this type. |
| index | The entire index is scanned |
| all | A full table scan is required |

# Performance Schema Profiling

SELECT event_id, sql_text FROM performance_schema.events_statements_history_long WHERE sql_text like ‘%select * from city%’;

```
+----------+--------------------+
| event_id | sql_text           |
+----------+--------------------+
|       54 | select * from city |
+----------+--------------------+
```





PERCONA

# Performance Schema Profiling cont.

use performance_schema;

```
SELECT
  event_name AS Stage,
  round(timer_wait/
    pow(10,12),6) AS Duration
FROM
events_stages_history_long
WHERE nesting_event_id = 54;
```

| Stage | Duration |
|---|---|
| stage/sql/starting | 0.000098 |
| stage/sql/checking permissions | 0.000001 |
| stage/sql/Opening tables | 0.000029 |
| stage/sql/init | 0.000007 |
| stage/sql/System lock | 0.000005 |
| stage/sql/optimizing | 0.000001 |
| stage/sql/statistics | 0.000014 |
| stage/sql/preparing | 0.000009 |
| stage/sql/executing | 0.000006 |
| stage/sql/Sending data | 0.000676 |
| stage/sql/end | 0.000002 |
| stage/sql/query end | 0.000007 |
| stage/sql/closing tables | 0.000010 |
| stage/sql/freeing items | 0.000047 |
| stage/sql/cleaning up | 0.000003 |

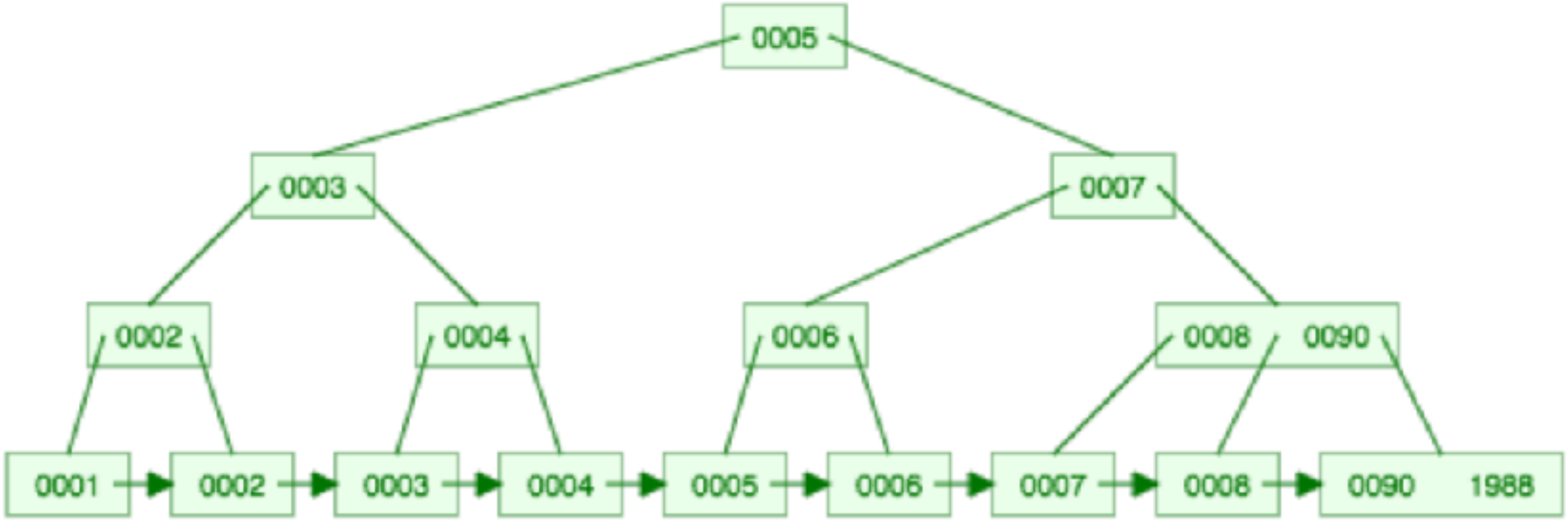
# Pt-query-digest

- Analyses MySQL queries from the slow, general, and binary logs
- Groups queries by fingerprint
- provides average runtime information

# Indexes

## Index concepts and limitations

# Balanced Tree Index

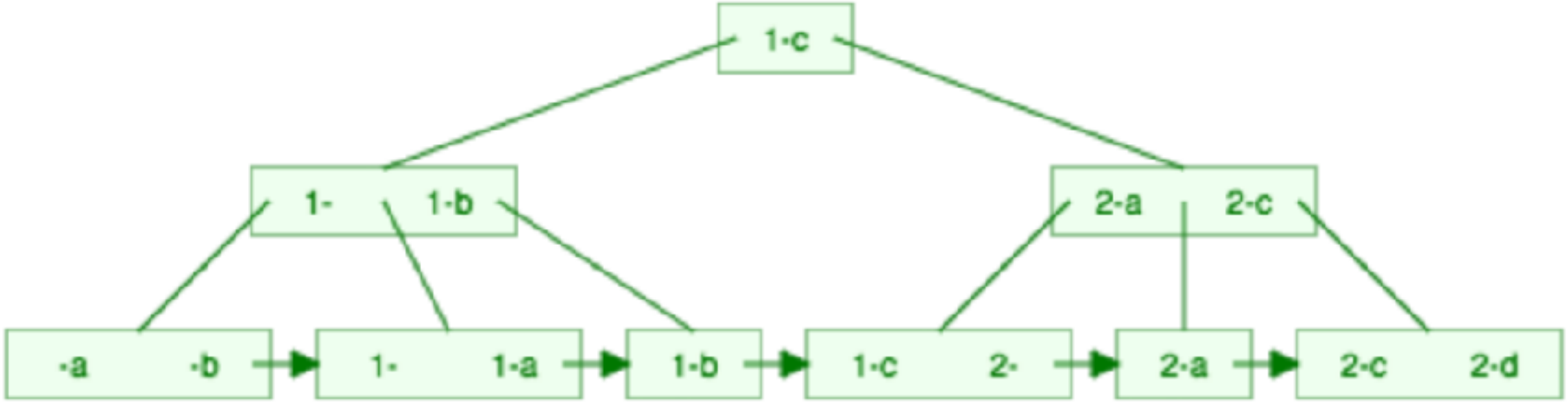







PERCONA

# Composite Balanced Tree Index








PERCONA

# Primary Key

## Requirements and constraints

- Field(s) MUST be UNIQUE
- NULL is not valid
- Only 1 per table
- Can be a single or multiple fields

## Controls how data is stored physically

## Is utilized as part of ALL secondary indexes

# Index Impact

SELECT SQL_NO_CACHE count(1) FROM sakila.rental WHERE last_update>'2006-02-18 00:00:00'\G

| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | SIMPLE | rental | NULL | ALL | NULL | NULL | NULL | NULL | 16005 | 33.33 | Using where |

```
# Query_time distribution
#   1us
#  10us
# 100us
#   1ms  ##################################################################
#  10ms  ########
# 100ms
#    1s
#  10s+
```





PERCONA

# Index Impact cont

SELECT count(1) FROM sakila.rental WHERE last_update>'2006-02-18 00:00:00'\G

| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | SIMPLE | rental | NULL | range | last_update | last_update | 4 | NULL | 1 | 100.00 | Using index condition |

```
# Query_time distribution
#   1us
#  10us
# 100us  ##################################################################
#   1ms  #
#  10ms
# 100ms
#    1s
#  10s+
```





PERCONA

# Multiple Tables

SELECT f.rating,sum(p.amount) as income FROM film f JOIN inventory i ON i.film_id = f.film_id JOIN rental r ON r.inventory_id = i.inventory_id JOIN payment p ON p.rental_id=r.rental_id GROUP BY f.rating ORDER BY income DESC;

| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | SIMPLE | f | NULL | ALL | PRIMARY | NULL | NULL | NULL | 1000 | 100.00 | Using temporary; Using filesort |
| 1 | SIMPLE | i | NULL | ref | PRIMARY,idx_fk_film_id | idx_fk_film_id | 2 | sakila.f.film_id | 4 | 100.00 | Using index |
| 1 | SIMPLE | r | NULL | ref | PRIMARY,idx_fk_inventory_id | idx_fk_inventory_id | 3 | sakila.i.inventory_id | 3 | 100.00 | Using index |
| 1 | SIMPLE | p | NULL | ref | fk_payment_rental | fk_payment_rental | 5 | sakila.r.rental_id | 1 | 100.00 | NULL |

```
# Query_time distribution
#   1us
#  10us
# 100us
#   1ms
#  10ms  ##################################################################
# 100ms  ##
#     1s
#  10s+
```





PERCONA

# Multiple Tables

SELECT f.rating,sum(p.amount) as income FROM film f JOIN inventory i ON i.film_id = f.film_id JOIN rental r ON r.inventory_id = i.inventory_id JOIN payment p ON p.rental_id=r.rental_id GROUP BY f.rating ORDER BY income DESC;

| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | SIMPLE | f | NULL | index | PRIMARY,rating | rating | 2 | NULL | 1000 | 100.00 | Using index; Using temporary; Using filesort |
| 1 | SIMPLE | i | NULL | ref | PRIMARY,idx_fk_film_id | idx_fk_film_id | 2 | sakila.f.film_id | 4 | 100.00 | Using index |
| 1 | SIMPLE | r | NULL | ref | PRIMARY,idx_fk_inventory_id | idx_fk_inventory_id | 3 | sakila.i.inventory_id | 3 | 100.00 | Using index |
| 1 | SIMPLE | p | NULL | ref | fk_payment_rental | fk_payment_rental | 5 | sakila.r.rental_id | 1 | 100.00 | NULL |

```
# Query_time distribution
#   1us
#  10us
# 100us
#   1ms
#  10ms  ##################################################################
# 100ms  #
#     1s
#  10s+
```





PERCONA

# Index Limitations

## Functions

- Aggregates
  - *SUM,AVG*
  - *Does not include MIN,MAX*
- Multiple Range scans
  - *multiple OR statements*
  - *multiple less than  or greater than*
- Character Set Differences
- Data Type Differences

# Multiple Range Scans

SELECT count(1) FROM rental WHERE customer_id=148 AND return_date BETWEEN '2005-08-29 00:00:00' AND '2005-09-05 23:59:59' AND last_update BETWEEN '2006-02-15 00:00:00' AND '2006-02-15 23:59:59';

| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | SIMPLE | rental | NULL | ref | idx_fk_customer_id,last_update | idx_fk_customer_id | 2 | const | 46 | 5.55 | Using where |

Add a composite index on customer_id,return_date,last_update

| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | SIMPLE | rental | NULL | range | idx_fk_customer_id,last_update,c_r_l | c_r_l | 12 | NULL | 4 | 50.00 | Using where; Using index |





PERCONA

# Multiple Range Scans

```
EXPLAIN: {
  "query_block": {
    "select_id": 1,
    "cost_info": {
      "query_cost": "1.81"
    },
    "table": {
      "table_name": "rental",
      "access_type": "range",
      "possible_keys": [
        "idx_fk_customer_id",
        "last_update",
        "c_r",
        "c_l",
        "c_r_l"
      ],
      "key": "c_r_l",
      "used_key_parts": [
        "customer_id",
        "return_date"
      ],
      "key_length": "12",
      "rows_examined_per_scan": 4,
      "rows_produced_per_join": 1,
      "filtered": "50.00",
      "using_index": true,
      "cost_info": {
        "read_cost": "1.62",
        "eval_cost": "0.20",
        "prefix_cost": "1.82",
        "data_read_per_join": "63"
      },
      "used_columns": [
        "customer_id",
        "return_date",
        "last_update"
      ],
      "attached_condition": "((`sakila`.`rental`.`customer_id` = 148) and (`sakila`.`rental`.`return_date` between '2005-08-29 00:00:00' and '2005-09-05 23:59:59') and (`sakila`.`r
ental`.`last_update` between '2006-02-15 00:00:00' and '2006-02-15 23:59:59'))"
    }
  }
}
```





PERCONA

# Multiple Range Scans

SELECT COUNT(1) FROM rental WHERE customer_id=148 AND (return_date BETWEEN '2005-08-29 00:00:00' AND '2005-09-05 23:59:59' OR last_update BETWEEN '2006-02-15 00:00:00' AND '2006-02-15 23:59:59');


```
EXPLAIN: {
  "query_block": {
    "select_id": 1,
    "cost_info": {
      "query_cost": "5.65"
    },
    "table": {
      "table_name": "rental",
      "access_type": "ref",
      "possible_keys": [
        "idx_fk_customer_id",
        "last_update",
        "c_r_l"
      ],
      "key": "c_r_l",
      "used_key_parts": [
        "customer_id"
      ],
      "key_length": "2",
      "ref": [
        "const"
      ],
      "rows_examined_per_scan": 46,
      "rows_produced_per_join": 9,
      "filtered": "20.00",
      "using_index": true,
      "cost_info": {
        "read_cost": "1.05",
        "eval_cost": "0.97",
        "prefix_cost": "5.65",
        "data_read_per_join": "288"
      },
      "used_columns": [
        "customer_id",
        "return_date",
        "last_update"
      ],
      "attached_condition": "((`sakila`.`rental`.`return_date` between '2005-08-29 00:00:00' and '2005-09-05 23:59:59') or (`sakila`.`rental`.`last_update` between '2006-02-15 00:00:00' and
'2006-02-15 23:59:59'))"
    }
  }
}
```






PERCONA

# Multiple Range Scans

SELECT COUNT(1) FROM rental WHERE customer_id=148 AND return_date BETWEEN '2005-08-29 00:00:00’ AND '2005-09-05 23:59:59'

UNION

SELECT COUNT(1) FROM rental WHERE customer_id=148 AND last_update BETWEEN '2006-02-15 00:00:00' AND '2006-02-15 23:59:59';

| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | PRIMARY | rental | NULL | range | idx_fk_customer_id,c_r_1 | c_r_1 | 8 | NULL | 4 | 100.00 | Using where; Using index |
| 2 | UNION | rental | NULL | ref | idx_fk_customer_id,last_update,c_r_1 | c_r_1 | 2 | const | 46 | 50.00 | Using where; Using index |
| NULL | UNION RESULT | <union1,2> | NULL | ALL | NULL | NULL | NULL | NULL | NULL | NULL | Using temporary |

Indexes are read left to right so a new index is needed on customer_id,last_update





PERCONA

# Multiple Range Scans

| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | PRIMARY | rental | NULL | range | idx_fk_customer_id,c_r,c_l | c_r | 8 | NULL | 4 | 100.00 | Using where; Using index |
| 2 | UNION | rental | NULL | range | idx_fk_customer_id,last_update,c_r,c_l | c_l | 6 | NULL | 46 | 100.00 | Using where; Using index |
| NULL | UNION RESULT | <union1,2> | NULL | ALL | NULL | NULL | NULL | NULL | NULL | NULL | Using temporary |

```
# Query_time distribution
#   1us
#  10us
# 100us  ##################################################################
#   1ms  ####
#  10ms
# 100ms
#    1s
#  10s+
```

OR

UNION

```
# Query_time distribution
#   1us
#  10us
# 100us  ##################################################################
#   1ms  ####
#  10ms
# 100ms
#    1s
#  10s+
```





PERCONA

# References

https://dev.mysql.com/doc/sakila/en/sakila-installation.html
https://dev.mysql.com/doc/refman/5.7/en/explain-output.html#explain-join-types
https://dev.mysql.com/doc/mysql-perfschema-excerpt/5.7/en/performance-schema-query-profiling.html
https://dev.mysql.com/doc/refman/5.7/en/general-thread-states.html
https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html
https://www.percona.com/software/database-tools/percona-toolkit
https://bugs.mysql.com/bug.php?id=83062