



ORACLE®

Getting the Best MySQL Performance in Your Products: Part 3, Query Tuning

Alexander Rubin

Principle Consultant MySQL

Copyright Oracle 2010

About MySQL

- Founded, first release in 1995
- Acquired by Sun in February 2008
- Acquired by Oracle in January 2010
- #1 Most Popular Open Source Database
- MySQL 5.5 RC
- Market-leading customers

Oracle's Plans for MySQL

- Complete Oracle's stack
- MySQL Global Business Unit
 - Managed by Edward Screven, Chief Corporate Architect
- Invest in MySQL!
 - “Make MySQL a Better MySQL”
 - Develop, Promote and Support MySQL
 - Improve engineering, consulting, and support
 - MySQL Sunday at Oracle Open World
 - Leverage World-Wide, 24x7 Oracle Support
- MySQL Community Edition
 - Source and binary releases
 - GPL license

MySQL 5.1 to 5.5 (RC) Improvements

- InnoDB becomes default
- Improved Availability, Improved Usability
- Better Instrumentation / Diagnostics
- InnoDB & MySQL Performance Improvements
 - More than 10x Improvement in recovery times
- Sysbench Results:
 - Linux: MySQL 5.5 vs. 5.1 - Read Only = 200%
 - Linux: MySQL 5.5 vs. 5.1 – Read Write = 369%
 - Windows: MySQL 5.5 vs. 5.1 - Read Only = 538%
 - Windows: MySQL 5.5 vs. 5.1 - Read Write = 1561%

Industry-Leading Customers



Web / Web 2.0



OEMs / ISVs



On Demand, SaaS, Hosting



Telecommunications



Enterprise 2.0

Rely on MySQL

Contents

- Monitoring your queries
- Understanding query performance
- Indexes
- Temporary tables in MySQL
- GROUP BY operations
- ORDER BY operations
- Subqueries
- Q&A

How to deal with slow queries

- Find the candidates
 - Slow query
 - Benchmark
- Profile/Explain
- Fix queries
 - Add indexes
 - Re-write queries
- = Better performance!

Slow Query Log

- Contains text of long running queries
 - log queries executing longer than **long_query_time** server variable (in seconds, but supports microseconds resolution when logging to file)
- Helps identify candidates for query optimization
- Enabling log
 - **--log-slow-queries** or **--log-slow-queries=file_name**
- Can log non-indexed queries
 - **--log-queries-not-using-indexes**
- Written to log file or table
 - **slow_log** table in **mysql** database
- Use **--log-short-format** option for less verbose logging

Slow Query Log

Enable (mysql 5.1)

```
mysql> set global long_query_time = 0.5;
```

```
mysql> set global slow_query_log = 1;
```

```
mysql> set global slow_query_log_file =  
    'mysql_slow.log';
```

Process Slow Query Log 1

- Slow query example

```
Tcp port: 3306  Unix socket: /data/mysql.sock
```

```
Time                Id Command      Argument
```

```
# Time: 101115  7:08:21
```

```
# User@Host: root[root] @ localhost [127.0.0.1]
```

```
# Query_time: 29.308532  Lock_time: 0.000219  Rows_sent: 0
```

```
  Rows_examined: 14465
```

```
select max(LOCK_VERSION) from a;
```

- Process the whole file and find most frequent queries

```
$ mysqldumpslow -s c mysql_slow.log >mysql_slow.log.c
```

```
Reading mysql slow query log from /db1/data3/mysql_slow.log
```

Process Slow Query Log 2

- Find top 10 slowest queries

```
-bash-3.00$ mysqldumpslow -s t -n 10 mysql_slow.log
Reading mysql slow query log from /db1/data3/mysql_slow.log
Count: 1  Time=1148.99s (1148s)  Lock=0.00s (0s)  Rows=0.0
(0) ,
insert into a select * from b

Count: 37  Time=2.28s (84s)  Lock=0.11s (4s)  Rows=0.0 (0) ,
Update a set CONTENT_BINARY = 'S' where ID = 3874

Count: 1  Time=29.31s (29s)  Lock=0.00s (0s)  Rows=0.0 (0) ,
select max(LOCK_VERSION) from b

...
```



Main query performance problems

- Full table scans (no index)
- Temporary tables
- Filesort

Using **EXPLAIN** to Analyze Queries

- Find out how the query optimizer would improve a **SELECT** query
- Useful optimizer information
 - Points out need to index
 - Finds out if optimizer is using existing indexes
 - Can help qualify query rewrites

Full table scan

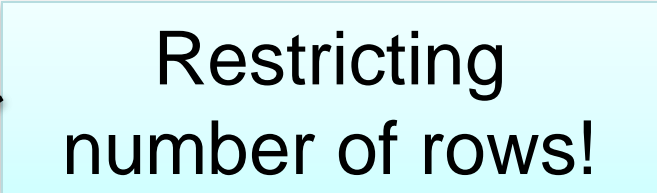
```
mysql> EXPLAIN select * from City where Name = 'London'\G
***** 1. row
```

```
      id: 1
select_type: SIMPLE
      table: City
      type: ALL
possible_keys: NULL
      key: NULL
key_len: NULL
      ref: NULL
      rows: 4079
Extra: Using where
```

Adding index to fix the query

```
mysql> alter table City add key (Name);  
Query OK, 4079 rows affected (0.02 sec)  
Records: 4079 Duplicates: 0 Warnings: 0
```

```
mysql> explain select * from City where Name = 'London'\G  
***** 1. row  
*****  
      id: 1  
select_type: SIMPLE  
      table: City  
      type: ref  
possible_keys: Name  
      key: Name  
      key_len: 35  
      ref: const  
      rows: 1  
      Extra: Using where  
1 row in set (0.00 sec)
```





Composite Indexes in MySQL

Composite Indexes

- MySQL choose 1 (best) index per table
 - With some exceptions...
- More unique values the better
 - Do not index status, gender, etc
- Order of fields inside index matters
 - (in most cases)
- **“Where region = ‘US’ and date_added>’ 2010-05-01’ “**
 - Index on (region, date_added) preferred

Composite Indexes

- `“Where region = ‘US’ and date_added>’ 2010-05-01’ “`
- Index (region, date_added):
 1. MySQL will “jump” to index leaf where Region=‘US’
 2. Scan date_added range starting with the leaf
- Constant + range: put constant first, range second



GROUP BY queries

GROUP BY and Temporary Tables

- How many cities in each country:

```
mysql> explain select CountryCode, count(*) from City
      group by CountryCode\G
```

```
***** 1. row
```

```
      id: 1
```

```
      select_type: SIMPLE
```

```
      table: City
```

```
      type: ALL
```

```
possible_keys: NULL
```

```
      key: NULL
```

```
      key_len: NULL
```

```
      ref: NULL
```

```
      rows: 4079
```

```
      Extra: Using temporary; Using filesort
```

```
1 row in set (0.00 sec)
```

Temporary tables
are slow!



Temporary Tables: Theory

Temporary Tables

- Main performance issues
- MySQL can create temporary tables when query uses
 - GROUP BY
 - Range + ORDER BY
 - Some other expressions
- 2 types of temporary tables
 - MEMORY
 - On-disk

Temporary Tables

- First, MySQL tries to create temporary table in memory
 - **tmp_table_size**
 - maximum size for in Memory temporary tables
 - **max_heap_table_size**
 - Sets the maximum size for **MEMORY** tables
- If (tmp_table > tmp_table_size OR
tmp_table > max_heap_table_size)
- { convert to MyISAM temporary table on disk }

Temporary Tables

- MEMORY engine does not support BLOB/TEXT
- `select blob_field from table group by field1`
- `select concat(...string>512 chars) group by field1`
 - Create on-disk temporary table right away

Temporary Tables: Profiling

- Watch:
 - **Created_tmp_tables** – number of temporary table MySQL created in both *RAM and DISK*
 - **Created_tmp_disk_tables** - number of temporary table MySQL created on *DISK*

```
mysql> show session status like 'created%';
+-----+-----+
| Variable_name          | Value |
+-----+-----+
| Created_tmp_disk_tables | 1     |
| Created_tmp_files      | 0     |
| Created_tmp_tables     | 10    |
+-----+-----+
3 rows in set (0.00 sec)
```



Temporary Tables: Practice

Air Traffic statistics table for tests

5M rows, ~2G in size

```
CREATE TABLE `ontime_2010` (  
  `YearD` int(11) DEFAULT NULL,  
  `MonthD` tinyint(4) DEFAULT NULL,  
  `DayofMonth` tinyint(4) DEFAULT NULL,  
  `DayOfWeek` tinyint(4) DEFAULT NULL,  
  `Carrier` char(2) DEFAULT NULL,  
  `Origin` char(5) DEFAULT NULL,  
  `DepDelayMinutes` int(11) DEFAULT NULL,  
  ...  
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

http://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236&DB_Short_Name=On-Time

GROUP BY Query example

- Find maximum delay for flights on Sunday
- Group by airline

```
select max(DepDelayMinutes) ,  
       carrier, dayofweek  
from ontime_2010  
where dayofweek = 7  
group by Carrier, dayofweek
```

GROUP BY Query example

```
select max(DepDelayMinutes), carrier, dayofweek
from ontime_2010
where dayofweek = 7
group by Carrier, dayofweek
```

type: ALL

possible_keys: NULL

key: NULL

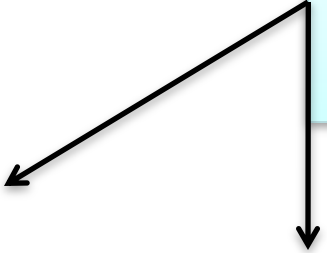
key_len: NULL

ref: NULL

rows: 4833086

Extra: Using where; Using temporary; Using
filesort

Full Table scan!
Temporary table!



Fixing full table scan: part 1

```
mysql> alter table ontime_2010 add key (dayofweek);
```

```
mysql> explain select max(DepDelayMinutes), Carrier,  
    dayofweek from ontime_2010  
where dayofweek =7 group by Carrier, dayofweek\G
```

```
    type: ref  
possible_keys: DayOfWeek  
    key: DayOfWeek  
key_len: 2  
    ref: const  
rows: 817258
```

Many rows scanned!
Temporary table!

```
Extra: Using where; Using temporary; Using filesort
```

GROUP BY: Adding covered index

```
mysql> alter table ontime_2010 add key  
      covered(dayofweek, Carrier, DepDelayMinutes);
```

```
mysql> explain select max(DepDelayMinutes), Carrier,  
      dayofweek from ontime_2010  
where dayofweek =7 group by Carrier, dayofweek\G
```

...

```
possible_keys: DayOfWeek,covered
```

```
      key: covered
```

```
      key_len: 2
```

```
      ref: const
```

```
      rows: 905138
```

```
      Extra: Using where; Using index
```

No temporary table!



Where covered index is not good enough...

```
mysql> explain select max(DepDelayMinutes), Carrier,  
    dayofweek from ontime_2010  
where dayofweek > 3  
group by Carrier,  
dayofweek\G  
...
```

```
    type: range  
possible_keys: covered  
    key: covered  
key_len: 2  
    ref: NULL  
    rows: 2441781  
Extra: Using where; Using index; Using  
temporary; Using filesort
```



GROUP BY: Loose index scan

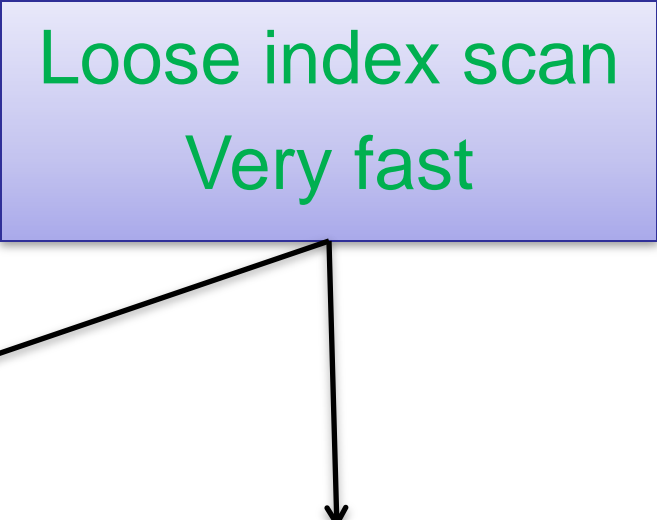
- Loose index scan:
 - access method, considers only a fraction of the keys in an index
- Following rules apply:
 - The query is over a single table.
 - The GROUP BY names only columns that form a leftmost prefix of the index and no other columns.
 - The only aggregate functions used in the select list (if any) are MIN() and MAX(), same column

Loose index scan example

```
mysql> alter table ontime_2010 add key lis1(Carrier,  
      dayofweek, DepDelayMinutes);
```

```
mysql> explain select max(DepDelayMinutes), Carrier,  
      dayofweek from ontime_2010 where dayofweek > 3 group  
      by Carrier, dayofweek \G
```

```
      ...  
table: ontime_2010  
      type: range  
possible_keys: NULL  
      key: lis1  
key_len: 5  
      ref: NULL  
      rows: 201  
Extra: Using where; Using index for group-by
```



Loose index scan
Very fast

GROUP BY: Tight index scan

- Tight index scan:
 - full index scan or a range index scan
- Can work if loose index scan can't be used
- Allow to scan index and AVOID creating tmp table
- The grouping operation is performed only after all keys that satisfy the range conditions have been found.

Loose index scan vs. tight index scan

- Benchmark: ontime_2010 table
 - 5M rows, data: 1.7G, index: 150M

```
CREATE TABLE `ontime_2010` (  
  `YearD` int(11) DEFAULT NULL,  
  `MonthD` tinyint(4) DEFAULT NULL,  
  `DayofMonth` tinyint(4) DEFAULT NULL,  
  `DayOfWeek` tinyint(4) DEFAULT NULL,  
  `Carrier` char(2) DEFAULT NULL,  
  `Origin` char(5) DEFAULT NULL,  
  `DepDelayMinutes` int(11) DEFAULT NULL,  
  ...  
  KEY `lis1` (`Carrier`, `DayOfWeek`, `DepDelayMinutes`),  
  KEY `covered` (`DayOfWeek`, `Carrier`, `DepDelayMinutes`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

[http://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236
&DB_Short_Name=On-Time](http://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236&DB_Short_Name=On-Time)

Loose index scan vs. tight index scan

- Loose index scan

```
mysql> explain select max(DepDelayMinutes), Carrier, dayofweek from
ontime_2010 where dayofweek = 3 group by Carrier, dayofweek
limit 10\G
```

```
table: ontime_2010
type: range
possible_keys: covered
key: lis1
key_len: 5
rows: 198
```

```
`lis1` (`Carrier`, `DayOfWeek`,
`DepDelayMinutes`)
```

Extra: Using where; **Using index for group-by**

```
mysql> select ...
```

```
+-----+-----+-----+
| max(DepDelayMinutes) | Carrier | dayofweek |
+-----+-----+-----+
|          1184 | 9E      |          3 |
...
+-----+-----+-----+
10 rows in set (0.00 sec)
```

Loose index scan vs. tight index scan

- Tight index scan

```
mysql> explain select max(DepDelayMinutes), Carrier, dayofweek from
ontime_2010 use index (covered) where dayofweek = 3 group by
Carrier, dayofweek limit 10\G
```

```
table: ontime_2010
type: ref
key: covered
key_len: 2
ref: const
rows: 2302412
Extra: Using where; Using index
```

``covered` (`DayOfWeek`,
`Carrier`, `DepDelayMinutes`)`

1. Found all keys
2. Perform group

```
mysql> select ...
```

```
+-----+-----+-----+
| max(DepDelayMinutes) | Carrier | dayofweek |
+-----+-----+-----+
|                1184 | 9E      |          3 |
```

...

```
10 rows in set (0.31 sec)
```

Where loose index scan is not supported

- AVG() – loose index scan is not supported
- Range scan – have to create temporary table

```
mysql> explain select AVG(DepDelayMinutes), Carrier, dayofweek from
  ontime_2010 where dayofweek > 3 group by Carrier, dayofweek limit
  10\G
```

```
table: ontime_2010
type: range
key: covered
key_len: 2
ref: NULL
rows: 2416543
```

1. No loose index scan
2. Filter by key
3. Group by filesort

```
Extra: Using where; Using index; Using temporary; Using filesort
```

```
mysql> select ...
```

max(DepDelayMinutes)	Carrier	dayofweek
730	9E	4

...

```
10 rows in set (5.38 sec)
```

Rewriting query using union

- where dayofweek in (6,7)
 - Convert range to ref:

```
select avg(DepDelayMinutes), Carrier, dayofweek  
from ontime_2010
```

```
where dayofweek = 6 group by Carrier, dayofweek  
UNION
```

```
select avg(DepDelayMinutes), Carrier, dayofweek  
from ontime_2010
```

```
where dayofweek = 7 group by Carrier, dayofweek
```


Rewriting query: explain plan

```
***** 1. row *****
  select_type: PRIMARY
    type: ref
      key: covered
    key_len: 2
      ref: const
    rows: 1191190
  Extra: Using where; Using index
***** 2. row *****
  select_type: UNION
    type: ref
      key: covered
    key_len: 2
      ref: const
    rows: 905138
  Extra: Using where; Using index
***** 3. row *****
  select_type: UNION RESULT
    table: <union1,2>
    type: ALL
      key: NULL
    key_len: NULL
      ref: NULL
    rows: NULL
  Extra:
```

Rewriting query: speed comparison

- Temporary table:

```
mysql> select avg(DepDelayMinutes), Carrier, dayofweek
        from ontime_2010 where dayofweek in (6,7) group by
        Carrier, dayofweek;
```

36 rows in set (3.64 sec)

- Union:

```
mysql> select avg(DepDelayMinutes), Carrier, dayofweek
        from ontime_2010 where dayofweek=6 group by Carrier,
        dayofweek union select avg(DepDelayMinutes),
        Carrier, dayofweek from ontime_2010 where dayofweek=7
        group by Carrier, dayofweek;
```

36 rows in set (1.67 sec)



ORDER BY and filesort

ORDER BY and filesort

- Find 10 cities in USA with the largest population

```
mysql> explain select district, name, population from  
City where CountryCode = 'USA' order by population  
desc limit 10\G
```

```
      table: City  
      type: ALL  
possible_keys: NULL  
      key: NULL  
key_len: NULL  
      ref: NULL  
      rows: 4079  
Extra: Using where; Using filesort
```

Fixing filesort: adding index

```
mysql> alter table City add key my_sort2 (CountryCode,  
population);
```

```
mysql> explain select district, name, population from  
City where CountryCode = 'USA' order by population  
desc limit 10\G
```

```
table: City  
type: ref  
key: my_sort2  
key_len: 3  
ref: const  
rows: 207  
Extra: Using where
```



No Filesort

Sorting and Limit

```
mysql> alter table ontime_2010 add key (DepDelayMinutes);  
Query OK, 0 rows affected (38.68 sec)
```

```
mysql> explain select * from ontime_2010  
where dayofweek in (6,7) order by DepDelayMinutes desc  
limit 10\G
```

```
           type: index  
possible_keys: DayOfWeek,covered  
           key: DepDelayMinutes  
key_len: 5  
           ref: NULL  
           rows: 24  
Extra: Using where
```

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

1. Index is sorted
2. Scan whole table in the order of the index
3. Filter results
4. Stop after finding 10 rows matching where



Subqueries optimizations

Subqueries

- Sub-query inside select

```
SELECT (SELECT s1 FROM t2) FROM t1;
```

- Sub-query inside where

```
SELECT * FROM t1 WHERE  
column1 in (SELECT column2 FROM t2);
```

- Sub-query in FROM and joins

```
SELECT sb1, sb2, sb3 FROM (SELECT s1 AS sb1, s2  
    AS sb2, s3*2 AS sb3 FROM t1) AS sb  
WHERE sb1 > 1;
```


Subqueries

- Sub-query inside select

```
SELECT (SELECT max(s1) FROM t2), ... FROM t1
where mydate>now();
10000 rows in set
```

- Will execute subquery SELECT max(s1) FROM t2 10000 times
- Can rewrite it:

```
SELECT max(s1) into @m FROM t2
SELECT @m, ... FROM t1 where ...
```

Subqueries

- Sub-query inside where

```
SELECT * FROM t1 WHERE
```

```
column1 in (SELECT column2 FROM t2) ;
```

- Will not use index on **column1**
- <http://bugs.mysql.com/bug.php?id=8139>
- Can rewrite query as join

Subqueries

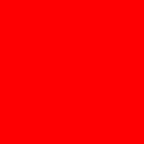
- Sub-query in FROM and joins

```
SELECT sb1, sb2, sb3 FROM (SELECT s1 AS sb1, s2  
    AS sb2, s3*2 AS sb3 FROM t1) AS sb  
WHERE sb1 > 1;
```

- MySQL will create a temporary table for (SELECT s1 AS sb1, s2 AS sb2, s3*2 AS sb3 FROM t1) with no indexes
- Rewrite as join

Resources and Q&A

- “Getting the Best MySQL Performance in Your Products: The Webinar Series”
 - **Part 1, The Fundamentals:**
<http://mysql.com/news-and-events/on-demand-webinars/display-od-552.html>
 - **Part 2, Beyond the Basics:**
<http://mysql.com/news-and-events/on-demand-webinars/display-od-567.html>
 - **Part 3, Query Tuning:** 1. You will receive e-mail with link;
2. Check webinar on-demand section under “Embedded” section:
<http://www.mysql.com/news-and-events/on-demand-webinars/>
- “InnoDB Enhancements and Roadmap” Webinar,
14 December, 9:00 am Pacific
 - <http://www.mysql.com/news-and-events/web-seminars/display-584.html>
- ISV / OEM Resources
 - <http://mysql.com/why-mysql/isv-oem-corner/>
- Questions?
 - <http://www.mysql.com/about/contact/sales.html?s=oem>
 - Phone: USA=+1-866-221-0634; Outside USA = +1-208-327-6494



The presentation is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.