

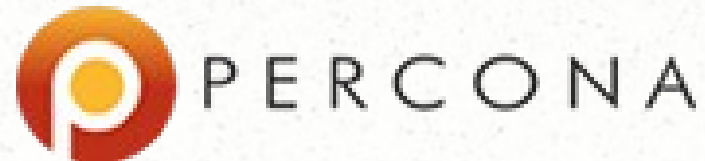
# *OKC MySQL Users Group*



# *OKC MySQL*

- Discuss topics about MySQL and related open source RDBMS
- Discuss complementary topics (big data, NoSQL, etc)
- Help to grow the local ecosystem through meetups and events

*Sponsored By...*



# *MySQL – Query Optimization*

“I mean, the query gives me the right answer, so why does it matter? My job is done!”  
- Way too many developers :)

## *What is it?*

- Human “questions” can generally be written in multiple forms in actual SQL
  - *Many* will even give the correct answer ;)
- Most queries will start out being poorly written and/or executed
- ORMs are notorious for writing “bad” queries
- Ways to optimize
  - Rewrite the query
  - Add indexes to the target tables

# *Optimization Basics*

- Examine as few rows as possible to get result set
- Read rows in sorted order
- Avoid creating temporary tables
  
- How do we do that?

**INDEXING!**

# *Indexing – High Level*

- An index in a database is the same in theory as the index in a book
- Which is faster?
  - Read every page and keep track of pages with X
  - Go to index, find X, jump to those pages
- Indexing works in the same way – shortcuts to data



As we've seen, you can do a whole 2 hour talk just on indexing – so that is outside the scope here...

# *Indexing – Basic Concepts*

- Columns you want to index
  - Those in where clause
  - Those being sorted/grouped
  - Those being joined
- You can create composite (multi-column) indexes
- MySQL uses composite indexes from Left → Right
- Indexes **DO** require space, so don't over index
- Composite indexes are often **BETTER** than several single indexes



## *How do I find queries?*

- Periodic review of production queries
- Review of all queries in pre-prod prior to release
- Developer review of queries while developing (this makes the above easier)
- And the number one way people find bad queries...
  - An outage in production!

## *Great, but how do I find them?*

- Slow query log
  - For historical review
- SHOW FULL PROCESSLIST
  - To find slow queries running now
  - (i.e. site is down and db is crawling)
- Please don't use the general log
  - Less info than slow log, much less useful

# *The Slow Log*

- This is the best tool for finding slow queries
- `long_query_time` defines threshold for queries to be reported
  - Note – this can be set to 0 to capture **all** queries
- Wealth of information
  - Rows examined vs rows returned
  - Execution time
  - Execution metadata (filesort, etc)
  - Percona Server offers additional metrics
- Numerous tools to parse the log
  - `pt-query-digest` is most used

## *I found one! Now what?!*

```
SELECT * FROM foo WHERE user_id = 1 ORDER  
BY date_created DESC
```

- Query takes forever to run
- Seems super easy since it should only return one row!
- First things first – can somebody tell me in English what that query is doing?

# *First Step...*

- Run EXPLAIN

```
mysql> EXPLAIN SELECT * FROM foo WHERE user_id = 1 ORDER BY date_created
DESC\G
***** 1. row *****
      id: 1
  select_type: SIMPLE
        table: foo
         type: ALL
possible_keys: NULL
          key: NULL
       key_len: NULL
         ref: NULL
        rows: 24
   Extra: Using where; Using filesort
1 row in set (0.00 sec)
```

## *Second Step...*

- Check the table structure (primarily indexes)

```
mysql> SHOW CREATE TABLE foo\G
***** 1. row *****
      Table: foo
Create Table: CREATE TABLE `foo` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) DEFAULT NULL,
  `data` varchar(255) DEFAULT NULL,
  `date_created` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=29 DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

## *Next...*

- Determine what index is best (won't always be perfect)
- Alter the table (you do have a test environment, right??)
- Re-run the query with explain
- Call it a day!

**Participation time!!**



# *To the VM we go!*

Let's work through the process and fix this:

```
SELECT * FROM foo  
WHERE user_id = 1  
ORDER BY date_created DESC
```



# *More than one way to skin a cat\**

- EXPLAIN – we've talked about this, but good first step
- Handler Operations
  - `FLUSH STATUS`
  - Run query
  - `SHOW STATUS LIKE 'ha%'`
  - Do this before/after any alters
- Query Profiling
  - `SET profiling = "ON"`
  - Run query
  - `SHOW PROFILE FOR QUERY 1`
  - Rinse/repeat

\* Please note, no animals were harmed during the making of this powerpoint!

*That is entirely too much work...*

While finding problem queries can be tedious, there are tools to make it much easier

... and queue shameless pitch now...

# Percona Cloud Tools!

The screenshot shows a web browser window displaying the Percona Cloud Tools interface. The URL is <https://cloud.percona.com/query-analytics/report>. The page header includes the Percona logo and navigation tabs for Performance, Metrics, Queries, Configure, and Admin. The 'Queries' tab is active, showing a report for host 'webapp01' from Dec 10, 2014, 10:25 to 11:25. The report is sorted by SUM and MAX. The main content area displays a 'Query Profile' table with columns for Rank, Query (distilled), Query ID, Queries, QPS, Load, Load %, Total Time, Avg Time, 95%, and Max Time. The table lists several queries, including a 'SHOW GLOBAL STATUS' query which is the most frequent.

Organization: okcmysql.org | User: michael.benshoof@percona.com

Host: webapp01 | Time Range: 1h 1d 1w 1m 3m 1y

Performance | Metrics | **Queries** | Configure | Admin

Dec 10 2014, 10:25 to 11:25 | Sort by: SUM MAX | Query Status: [OK] [WARN] [ERR] | Tags: [Filter]

### webapp01 [System Info](#)

### Query Profile

Rank	Query (distilled)	Query ID	Queries	QPS	Load	Load %	Total Time	Avg Time	95%	Max Time
	<b>Total</b>		5.00K	1.39	0.002	100%	7.35s	313.00µs	493.00µs	47.84ms
1	✓ SHOW GLOBAL STATUS (1)	B90978440CC11CC7	3.54K	0.98	0.002	92.75%	6.81s	1.92ms	3.56ms	47.84ms
2	➔ DELETE wp_options	AA8DB00350539EB6	63.00	0.02	0	2.6%	192.39ms	3.07ms	6.86ms	9.80ms
3	➔ INSERT UPDATE wp_options	8AE5000CAF43D53F	62.00	0.02	0	2.01%	148.71ms	2.41ms	3.18ms	3.42ms
4	➔ UPDATE wp_options	94350EA2AB8AAC34	22.00	0.01	0	0.78%	58.48ms	2.66ms	3.53ms	3.69ms
5	✓ SELECT wp_options (1)	7AEDF19FDD3A33F1	421.00	0.12	0	0.54%	40.67ms	95.00µs	167.00µs	263.00µs
6	⚠ SELECT wp_options	92F3B1B361FB055B	16.00	0.00	0	0.15%	10.30ms	663.00µs	673.00µs	858.00µs

*Let's take it for a spin...*

... assuming all of you promise not to hack okcmysql.org now  
that you can see some of my database structure!

# *Questions?*

(I know you have them, that's why you came today...  
so don't be shy)

*Thanks for coming!*

Website: <http://okcmysql.org>

Twitter: @okcmysql

Email: [mike@okcmysql.org](mailto:mike@okcmysql.org)