



Barry Austin  
doBoard, Inc.  
Interactive Strategies

# Optimizing MySQL for Web Applications

---

# The Big Picture

- Web apps can be slow for many reasons
- Some causes can be anticipated and avoided during design
- In the real world, causes often aren't apparent until the app is live, sometimes not until after the app has logged serious mileage
- Very often the database is the culprit. Or rather, the way the database is set up and used is the culprit.



# Design & Planning

- MySQL can be powerful and fast – use it!
- Think in terms of sets and relationships, not procedural program flow

# Design & Planning

- Prefer normalized schema design
  - Alternative to "Spreadsheet Syndrome"
  - Avoids data redundancies and anomalies
  - Allows more efficient indexing
  - But don't take it too far
  - <http://dev.mysql.com/tech-resources/articles/intro-to-normalization.html>

# Design & Planning

- Set up indexes

```
CREATE TABLE mytable (  
    id int(11) unsigned NOT NULL auto_increment,  
    things varchar(50),  
    stuff varchar(100),  
    PRIMARY KEY (id),  
    KEY (things)  
);
```

```
CREATE INDEX stuff ON mytable (stuff);
```

# Design & Planning

- JOIN is your friend
  - Makes normalized data access work
- JOIN-ed data types should match exactly to allow indexing to work properly

```
SELECT mytable.id, stuff, other_things  
FROM mytable JOIN myothertable  
ON mytable.id = myothertable.mytable_id;
```

# Design & Planning

- Choose optimal data types
- Pick appropriate storage engines
- Size tables for speed where necessary
- Query for what you need and no more



# App is Running – Now What?

- Profile

- Profile

- Profile



# Profiling MySQL

- Slow query log
  - `mysqld --log-slow-queries[=file_name]`
  - `long_query_time`
  - `mysqldumpslow`
- Index variant
  - `--log-queries-not-using-indexes[=file_name]`



# Profiling MySQL

- General query log
  - `mysqld --log[=file_name]`
  - `general_log`
  - `general_log_file`

# Profiling MySQL

- MySQL Query Profiler
  - <http://dev.mysql.com/tech-resources/articles/using-new-query-profiler.html>

# Profiling MySQL

- SHOW FULL PROCESSLIST
  - "too many connections"
  - Displays status of MySQL threads
  - <http://dev.mysql.com/doc/refman/5.0/en/show-processlist.html>



# Fixing Slow Queries

- Check scope
- Too many fields? Too many rows?
- `SELECT * FROM mytable`
- `SELECT a, b FROM mytable`
- `SELECT a, b FROM mytable WHERE c < 5`
- `SELECT a, b FROM mytable WHERE c < 5  
LIMIT 1`

# Fixing Slow Queries

- Check indexes
- SHOW INDEX FROM mytable;

```
mysql> SHOW INDEX FROM mytable;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment
mytable	0	PRIMARY	1	id	A	1	NULL	NULL		BTREE	
mytable	1	things	1	things	A	NULL	NULL	NULL	YES	BTREE	
mytable	1	stuff	1	stuff	A	NULL	NULL	NULL	YES	BTREE	

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

# Fixing Slow Queries

- EXPLAIN
  - <http://dev.mysql.com/doc/refman/5.1/en/using-explain.html>
  - Also see Jay Pipes' slides linked at the end

```
mysql> EXPLAIN SELECT mytable.id, stuff, other_things
-> FROM mytable JOIN myothertable
-> ON mytable.id = myothertable.mytable_id;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	mytable	system	PRIMARY	NULL	NULL	NULL	1	
1	SIMPLE	myothertable	system	NULL	NULL	NULL	NULL	1	

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

# Fixing Slow Queries

- mysql\_explain\_log
  - <http://dev.mysql.com/doc/refman/5.0/en/mysql-explain-log.html>



# Refactor Inefficient Code

- Hit the database only as much as you need to
- When possible, use SQL instead of PHP to filter and manipulate your result set

# Refactor Database Design

- Normalize – JOIN
- Tweak data types
- Split rarely used rows/columns from frequently used ones

# Let MySQL Be Lazy

- MySQL query cache
  - Enable by setting `query_cache_size`, `query_cache_type`
- `SHOW VARIABLES LIKE 'have_query_cache';`
- `SHOW STATUS LIKE 'Qcache%';`
- Pay attention to query cache rules and limitations
  - <http://dev.mysql.com/doc/refman/5.1/en/query-cache-how.html>

# Let MySQL Be Lazy

- Page cache
- Partial page cache
- Intermediate data cache

# Tuning

- ANALYZE TABLE
  - Updates key distribution data for the benefit of the built-in query optimizer
  - Use sparingly – needed only when optimizer is making poor choices due to inaccurate data



# Tuning

- For MyISAM:
  - `key_buffer_size`
  - `table_cache`
- For InnoDB:
  - `innodb_buffer_pool_size`
  - `innodb_flush_log_at_trx_commit`
- For both:
  - `query_cache_size`
  - `thread_cache`
  - check preconfigured option files `my-*.cnf`

# Other Tricks

- Avoid data types that aren't MySQL-friendly
  - BLOBs, TEXT, ENUM, etc.
- Store temporary data in temporary places
  - Session
  - Filesystem
  - Temporary tables
  - Memory storage engine, memcached, APC
- Clean out stale data
- Throw hardware at it



# Resources

- <http://dev.mysql.com/doc/refman/5.1/en/query-speed.html>
- <http://www.jpipes.com/index.php?/archives/260-Slides-from-Drunken-Query-Master-and-Join-fu-Talks-at-ZendCon.html>