

# Developing modern applications with MySQL

MySQL Overview  
Effective MySQL Architecture and Design Practices  
Effective Software Development with MySQL  
Effective Web Site Operations  
Upcoming MySQL features for modern applications



Helsinki  
20 Aug



Stockholm  
21 Aug



Riga  
26 Aug



Tallinn  
27 Aug

@RonaldBradford



# Effective Software Development with MySQL

Ronald Bradford  
August 2014

@RonaldBradford  
#mysql #emea2014

EffectiveMySQL.com

2

Effective Software Development with MySQL

## INFORMATION

These slides can be found after this  
EMEA tour at

<http://effectiveMySQL.com>

See website for links and  
more information.

#mysql #emea2014

EffectiveMySQL.com

3

Effective Software Development with MySQL

## 25+ YEARS EXPERIENCE

“The greatest performance  
overhead in MySQL systems today  
is the lack of SQL skills and general  
development skills by the software  
creators.”

Ronald Bradford  
August 2014

#mysql #emea2014

EffectiveMySQL.com

4

## CHANGE IN ROLES/SKILLS

From when I started 25 years ago

### Historically

- \* System Architect(s)
- \* Senior Analyst(s)
- \* Developer(s)
- \* QA Tester(s)
- \* Database Administrator(s)
- \* System Administrator(s)

### Now

- \* Software Developer
- \* Software Engineer
- \* Coder
- \* Entrepreneur

## EFFECTIVE TIPS

1. SQL Basics
2. SQL Constructs to avoid
3. SQL Constructs to use
4. Use Transactions
5. Do Less Work

...

## EFFECTIVE TIPS

6. The problem with timezones
7. The lack of standards
8. Read the MySQL documentation

## SQL Basics

# COMMENT YOUR SQL

\* Add a C style comment to any/all SQL

```
SELECT /* MySQLMEA2014 Demo */ NOW();
+-----+
| NOW() |
+-----+
| 2014-08-09 14:01:18 |
+-----+

INSERT /* AddGameStats:42 */ INTO stats(counter) VALUES(1);
```

\* Enables DBA to more easily find SQL via tools like grep.

mysql -c

# ENABLE GENERAL LOG

\* For all developer/test environments

\* NOT FOR PRODUCTION

5.1+ syntax

```
[mysqld]
general-log=1
general-log-file=/mysql/log/general.log
```

Demonstrate use later

```
tail -f /mysql/log/general.log
...
15:01:18 6 Query SELECT /* MySQLMEA2014 Demo */ NOW()
15:02:09 6 Query INSERT /* AddGameStats:42 */ INTO stats(counter)
```

# WHY VIEW QUERIES?

\* Developer needs to find bad habits early

\* Row At a Time (RAT) processing Discuss in Point 5

\* The total queries to be executed

\* Turn all caching off (memcache, QC etc)

\* Learn about the Query Execution Plan (QEP) and how to appreciate EXPLAIN

\* Understand the true impact of frameworks

# FRAMEWORKS

## CLAIM

\* improve speed of development

\* Abstract need to know SQL

## REALITY

\* Undocumented cost to sub-optimal performance, especially with data persistence

# FUTURE PROOF SQL

- \* Always alias columns when using joins
- \* SELECT only required columns
- \* Define columns for INSERTs

# ALIAS COLUMNS

```
CREATE TABLE products (
  id      INT UNSIGNED NOT NULL,
  name    VARCHAR(100) NOT NULL,
  description TEXT NULL);

CREATE TABLE orders (
  id          INT UNSIGNED NOT NULL,
  product_id INT UNSIGNED NOT NULL,
  qty         TINYINT UNSIGNED NOT NULL);

SELECT product_id, description
FROM   orders o, products p
WHERE  o.product_id = p.id;
```

# ALIAS COLUMNS

```
ALTER TABLE orders ADD description TEXT NULL;

SELECT product_id, description
FROM   orders o, products p
WHERE  o.product_id = p.id;

ERROR 1052 (23000): Column 'description' in field list is ambiguous

SELECT o.product_id, p.description
FROM   orders o, products p
WHERE  o.product_id = p.id;
```

# FUTURE PROOF INSERT

```
CREATE TABLE countries (
  code    CHAR(2) NOT NULL PRIMARY KEY,
  name    VARCHAR(60) NOT NULL,
  capital VARCHAR(60) NOT NULL
);
```

```
INSERT INTO countries
VALUES ('AU', 'Australia', 'Canberra');
```

```
INSERT INTO countries (code, name, capital)
VALUES ('US', 'USA', 'Washington DC');
```

# FUTURE PROOF INSERT

```
CREATE TABLE countries (
  code CHAR(2) NOT NULL PRIMARY KEY,
  name VARCHAR(60) NOT NULL,
  capital VARCHAR(60) NOT NULL
);
ALTER TABLE countries
  ADD population INT UNSIGNED NULL;
```

```
INSERT INTO countries
VALUES ('AU', 'Australia', 'Canberra');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
```

```
INSERT INTO countries (code, name, capital)
VALUES ('CA', 'Canada', 'Ottawa');
```



# SQL LINT

\* valid SQL can produce invalid results

```
SELECT YEAR(created) AS yy, MONTH(created) AS mm,
COUNT(*) AS cnt, FORMAT(SUM(total),2) AS total
FROM sales;
```

sql\_mode helps to identify and correct

\* 1 row

```
SELECT YEAR(created) AS yy, MONTH(created) AS mm,
COUNT(*) AS cnt, FORMAT(SUM(total),2) AS total
FROM sales
GROUP BY YEAR(created), MONTH(created);
```

\* n rows

# SQL LINT

\* valid SQL can produce invalid results

```
SELECT o.id,o.state,o.zip
FROM sales_tax o
LEFT JOIN new_sales_tax n ON (o.state = n.state AND o.zip = n.zip)
AND n.id IS NULL;
```

\* 6,000 rows

AND is part of ON, not part of WHERE

```
SELECT o.id,o.state,o.zip
FROM sales_tax o
LEFT JOIN new_sales_tax n ON (o.state = n.state AND o.zip = n.zip)
WHERE n.id IS NULL;
```

\* 6 rows

# RECAP

## SQL Basics

- \* Comment your SQL
- \* Log, View and Review ALL SQL
- \* Future proof your SQL
- \* A valid SQL syntax is not always correct

# Non standard SQL

# MYSQL SYNTAX

- \* REPLACE
- \* DELAYED
- \* IGNORE
- \* LOW\_PRIORITY
- \* HIGH\_PRIORITY
- \* SHOW & DESC

# REPLACE

- \* If row exists UPDATE, if does not exist then INSERT e.g. UPSERT

## NOT THIS, BUT:

- \* Implemented as DELETE row, then INSERT

Also no REPLACE TRIGGER option

# DELAYED/IGNORE

- \* INSERT DELAYED
- \* INSERT IGNORE
  - \* Do not provide errors when errors occur
- \* LOW\_PRIORITY | HIGH\_PRIORITY
  - \* Changing locking strategy

Does not work with InnoDB

## SHOW

## \* SHOW

- \* 41 different options in 5.5
- \* Use INFORMATION\_SCHEMA when possible
  - \* 5.0 - 17 views      \* 5.6 - 58+ views
  - \* 5.1 - 27+ views    \* 5.7 - 59+ views
  - \* 5.5 - 39+ views

## MySQL SQL Constructs

## INFORMATION SCHEMA

*“INFORMATION\_SCHEMA is a database within each MySQL instance, the place that stores information about all the other databases that the MySQL server maintains. The INFORMATION\_SCHEMA database contains several read-only tables. They are actually views, not base tables, so there are no files associated with them, and you cannot set triggers on them. Also, there is no database directory with that name.”*

<http://dev.mysql.com/doc/refman/5.5/en/information-schema.html>

## MYSQL SYNTAX

## \* Multi-row INSERT

- \* VALUES (), (), ()....

```
TRUNCATE TABLE stats;
START TRANSACTION;
INSERT INTO stats(counter)
VALUES (1), (2), (3), (4), (5), (6), (7), (8), (9);
COMMIT;

SELECT * FROM stats;
```

# MYSQL SYNTAX

## \* LIMIT [n],[m]

```

SELECT counter FROM stats WHERE counter > 2 LIMIT 2;
+-----+
| counter |
+-----+
|        3 |
|        4 |
+-----+
2 rows in set (0.00 sec)
SELECT counter FROM stats WHERE counter > 2 LIMIT 3,2;
+-----+
| counter |
+-----+
|        6 |
|        7 |
+-----+
2 rows in set (0.00 sec)

```

# STRING MATCHING

```

SELECT Name,CountryCode FROM City
WHERE LOWER(name) LIKE 'bris%';
+-----+-----+
| Name      | CountryCode |
+-----+-----+
| Brisbane  | AUS         |
| Bristol   | GBR         |
+-----+-----+

```

No CASE function  
needed

```

SELECT Name,CountryCode FROM City
WHERE name LIKE 'bris%';
+-----+-----+
| Name      | CountryCode |
+-----+-----+
| Brisbane  | AUS         |
| Bristol   | GBR         |
+-----+-----+

```

<http://dev.mysql.com/doc/index-other.html>

# STRING MATCHING

- \* String comparison is case insensitive by default
- \* Character set collation is configurable at column, table, schema level
- \* MySQL supports 70+ collations for 30+ character sets.
- \* MySQL does not support function based indexes

# Transactions



# TRANSACTIONS

- \* The foundation of RDBMS
  - \* The basis of ACID
  - \* Required for business systems
- BUT**
- \* So often no used in MySQL systems

## WHY?

# AUTOCOMMIT

SESSION only  
scope until 5.5

- \* MySQL defaults with autocommit on
- \* Historical default storage engine (MyISAM) did not support transactions.
  - \* InnoDB existed in 3.23 (Since 2001)
- \* Many open source projects fail to use transactions (a necessary good practice)
- \* New developers have never learned what a transaction is. and why they are critical to RDBMS

# TRANSACTIONS

- \* The general log tells all

What not to see

Time	Id	Command	Argument
140803 14:50:04	1	Connect	msandbox@localhost on test
140803 14:50:52	1	Query	INSERT INTO countries VALUES ('AU','Australia','Canberra')
140803 14:51:11	1	Query	INSERT INTO countries (code, name, capital) VALUES ('US','USA','Washington DC')
140803 14:51:13	1	Quit	

Correct Use

Time	Id	Command	Argument
140803 14:53:03	2	Connect	msandbox@localhost on test
140803 14:53:22	2	Query	START TRANSACTION
140803 14:53:29	2	Query	INSERT INTO countries VALUES ('AU','Australia','Canberra')
140803 14:53:47	2	Query	INSERT INTO countries (code, name, capital) VALUES ('US','USA','Washington DC')
140803 14:53:50	2	Query	COMMIT
140803 14:53:54	2	Quit	

# TRANSACTIONS

```
SELECT variable_name,variable_value
FROM information_schema.global_status
WHERE variable_name IN ('com_insert','com_update',
'com_delete', 'com_commit','com_rollback');
+-----+-----+
| variable_name | variable_value |
+-----+-----+
| COM_COMMIT    | 2148778        |
| COM_DELETE    | 3614178        |
| COM_INSERT    | 68096778       |
| COM_ROLLBACK  | 85193          |
| COM_UPDATE    | 142914426      |
+-----+-----+
```

# TRANSACTIONS

MyISAM Tables

```
SELECT table_schema,
       SUM(data_length+index_length)/1024/1024 AS total_mb,
       SUM(data_length)/1024/1024 AS data_mb,
       SUM(index_length)/1024/1024 AS index_mb,
       COUNT(*) AS tables,
       CURDATE() AS today
FROM   information_schema.tables
WHERE  engine='MyISAM'
AND    table_schema NOT IN ('mysql','information_schema')
GROUP BY table_schema
ORDER BY 2 DESC;
```

table_schema	total_mb	data_mb	index_mb	tables
xxxxxxxx	1418.39929962	1412.66883087	5.73046875	2
xxxx	16.89165497	9.18071747	7.71093750	3
xx_wordpress	7.82143688	7.45424938	0.36718750	13
xxxxxxxx_drupal	2.17076111	1.86607361	0.30468750	62

# USE CORRECTLY

- \* Abuse of Transactions
  - \* One business function, 3 transactions
  - \* Start transaction, then programming logic (taking time to execute)
  - \* Additional SELECT's inside transaction
- \* Frameworks exhibit these patterns

# 25+ YEARS EXPERIENCE

“If your application (in-house, open-source, purchased) does not use transactions at all times, there will be far greater problems and long term increased cost of ownership.”

Ronald Bradford  
August 2014

# Do Less Work

# 25+ YEARS EXPERIENCE

“Every MySQL application executes more SQL than necessary. Reducing, removing and combining SQL queries will improve performance and increase throughput capacity.”

Ronald Bradford  
August 2014

# EXAMPLES

- \* Duplicate identical queries
- \* Repeating queries
- \* Queries that can be combined
- \* Queries that can be simplified
- \* Queries that can be removed

# RAT PROCESSING

```

SELECT name FROM firms WHERE id=727;
SELECT name FROM firms WHERE id=758;
SELECT name FROM firms WHERE id=857;
SELECT name FROM firms WHERE id=740;
SELECT name FROM firms WHERE id=849;
SELECT name FROM firms WHERE id=839;
SELECT name FROM firms WHERE id=847;
SELECT name FROM firms WHERE id=867;
SELECT name FROM firms WHERE id=829;
SELECT name FROM firms WHERE id=812;
SELECT name FROM firms WHERE id=868;
SELECT name FROM firms WHERE id=723;

```



Classic N+1 problem

```

SELECT id, name
FROM firms
WHERE id IN (723, 727, 740, 758, 812, 829, 839, 847, 849, 857, 867, 868);

```



# CAT PROCESSING

```

SET PROFILING=1;
SELECT ...
SHOW PROFILES;

```

Query_ID	Duration	Query
1	0.00030400	SELECT name FROM firms WHERE id=727;
2	0.00014400	SELECT name FROM firms WHERE id=758;
3	0.00014300	SELECT name FROM firms WHERE id=857;
4	0.00014000	SELECT name FROM firms WHERE id=740;
5	0.00012300	SELECT name FROM firms WHERE id=849;
6	0.00012200	SELECT name FROM firms WHERE id=839;
7	0.00011600	SELECT name FROM firms WHERE id=847;
8	0.00014300	SELECT name FROM firms WHERE id=867;
9	0.00013900	SELECT name FROM firms WHERE id=829;
10	0.00014000	SELECT name FROM firms WHERE id=812;
11	0.00012800	SELECT name FROM firms WHERE id=868;
12	0.00011700	SELECT name FROM firms WHERE id=723;
13	0.00031100	SELECT id, name FROM firms WHERE id IN (723, 727, 740, 758, 812, 829, 839, 847, 849, 857, 867, 868);

4X longer processing for every page load

txt	total_time
Sum Individual Queries	0.001311
Combined Query	0.000311

Instant DB Scalability

# SIMPLICITY

Before considering how do I fix this code/SQL, ask?

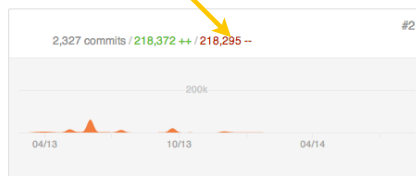
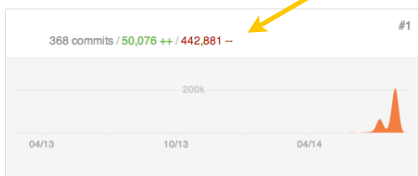
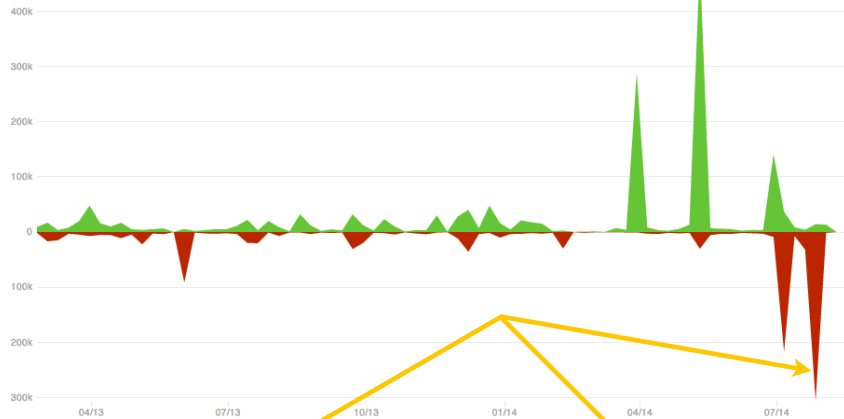
Can I remove or simplify this code/SQL?  
Does this process use this data?

# FRAMEWORK COST

**Drupal “Framework” + customization**

- \* 50 SQL statements to create a user
- \* Only 13 actual SQL statements required with sub-optimal design
- \* 4-5 SQL statements with optimal design

If this exists in the most fundamental core, it exists everywhere in code



# MySQL and Timezones

# TIMEZONE FUNCTIONS

- \* Will your users ever cross timezones?
- \* Will your application data ever support multiple timezones?
  - \* CONVERT\_TZ(date,from\_tz,to\_tz)
  
- \* Does not work by default

# TIMEZONES

- \* Timezone data is not loaded by default

```
mysql> SELECT COUNT(*) AS cnt FROM mysql.time_zone;
+-----+
| cnt |
+-----+
| 0 |
+-----+

mysql> SELECT NOW(),
-> CONVERT_TZ(NOW(), 'America/New_York', 'UTC') AS utc;
+-----+-----+
| NOW() | utc |
+-----+-----+
| 2014-08-02 17:05:18 | NULL |
+-----+-----+
```

<http://dev.mysql.com/doc/refman/5.6/en/date-and-time-functions.html>

# TIMEZONES

```
$ mysql_tzinfo_to_sql /usr/share/zoneinfo | \
mysql -u root mysql

mysql> SELECT COUNT(*) AS cnt FROM mysql.time_zone;
+-----+
| cnt |
+-----+
| 1738 |
+-----+

mysql> SELECT NOW(),
-> CONVERT_TZ(NOW(), 'America/New_York', 'UTC') AS utc;
+-----+-----+
| NOW() | utc |
+-----+-----+
| 2014-08-02 16:14:04 | 2014-08-02 20:14:04 |
+-----+-----+
```

<http://dev.mysql.com/doc/refman/5.6/en/mysql-tzinfo-to-sql.html>

# TIMEZONES

```
SELECT @@time_zone;
+-----+
| @@time_zone |
+-----+
| SYSTEM |
+-----+
```

Does not tell you the actual TZ

```
[mysql]
default_time_zone='America/Chicago';
```

```
SELECT @@time_zone;
+-----+
| @@time_zone |
+-----+
| America/Chicago |
+-----+
```

# MYSQL SYNTAX

\* Do not use system global in coding

```
SELECT @@system_time_zone;
+-----+
| @@system_time_zone |
+-----+
| EDT                 |
+-----+
```



- \* String value not always loaded
- \* Set at startup, does not change for daylight savings

```
SELECT CONVERT_TZ(NOW(),@@system_time_zone,'UTC') AS wrong,
       CONVERT_TZ(NOW(),@time_zone,'UTC') AS correct;
+-----+-----+-----+
| wrong | correct |
+-----+-----+-----+
| NULL  | 2014-08-05 02:07:18 |
+-----+-----+-----+
```

# Having Standards

# ABOUT STANDARDS

- \* This presentation is not about defining THE standard to use
- \* This is about stating you HAVE a standard that meets your needs, you USE it, and ENFORCE it
- \* Some recommendations...

# BE CONSISTENT

- \* Name columns consistently
  - \* column for date/time a row is created

```
created_date  DATETIME
datetime     DATETIME
created_at    DATETIME
created       TIMESTAMP
```

One data model, at least 4 variances

# PRIMARY KEY

- \* Name primary key unique across system

```
CREATE TABLE order (
  order_id INT UNSIGNED NOT NULL PRIMARY KEY,
  customer_id INT UNSIGNED NOT NULL,
```

```
CREATE TABLE customer (
  customer_id INT UNSIGNED NOT NULL PRIMARY KEY,
```

```
CREATE TABLE order (
  id INT UNSIGNED NOT NULL PRIMARY KEY,
  customer_id INT UNSIGNED NOT NULL,
```

```
CREATE TABLE customer (
  id INT UNSIGNED NOT NULL PRIMARY KEY,
```



# DOCUMENTATION

- \* Where is the data model?

# RESERVED WORDS

- \* Do not use reserved words as columns
  - \* datetime
  - \* date
  - \* from
  - \* to

MySQL allows use of reserved words using `backtick` syntax

# TABLE NAMES

- \* Name tables either singular or plural
  - \* Again, be consistent
- \* Professionally, singular wins because it is simpler
- \* order, order\_line, country
- \* orders, order[s]\_lines, countries

# Learn the Documentation

# DOCS

<http://dev.mysql.com/doc/>

General	Administrators	MySQL Enterprise	Developers & Functionality	Connectors & APIs	HA/Scalability
Tutorial	Installation & Upgrades	MySQL Enterprise Edition	= MySQL Workbench	Connectors and APIs	= HA/Scalability Guide
Server Administration	MySQL Yum Repository	MySQL Enterprise Monitor	Globalization	Connector/J	MySQL and DBBD
SQL Syntax	= MySQL Installer	MySQL Enterprise Backup	Optimization	Connector/ODBC	memcached
Storage Engines	= Security	MySQL Enterprise Security	Functions and Operators	Connector/Net	memcached with InnoDB
Server Option / Variable Reference	= Startup / Shutdown	MySQL Enterprise Audit	Views and Stored Programs	Connector/Python	MySQL and Virtualization
= Release Notes	= Backup and Recovery Overview	MySQL Thread Pool	Partitioning	PHP	MySQL Proxy
= MySQL Version Reference	= MySQL Utilities		Precision Math	C API	Replication
FAQs	= Linux/Unix		Information	Connector/C	Semiasynchronous

# What next?

# MORE

Today's Developers

- \* Developer should learn EXPLAIN
- \* Scrub/Subset your data
  - \* Developer should not use production data
- \* Actively remove unused tables/columns
- \* Do manual code deployments
- \* Enforce a QA step



## CONCLUDING TOP TIPS

- \* Do Less Work
- \* Ensure developers learn how to read and write SQL
  - \* Use Transactions
- \* Read the MySQL documentation

## TRIVIA TIP

- \* `--i-am-a-dummy`
  - \* Restricts UPDATE/DELETE statements
  - \* Alias for `--safe-updates`

## CONCLUSION

Copies of these slides can be found at

<http://effectiveMySQL.com>

[ronald@EffectiveMySQL.com](mailto:ronald@EffectiveMySQL.com)