EffectiveMySQL.com

Its all about Performance and Scalability

MySQL Best Practices

for DBAs and Developers

Volume I

Ronald Bradford
http://ronaldbradford.com

2011.04

AGENDA

- Essential MySQL configuration
- MySQL user security
- Improving your SQL
  - Capture
  - Analysis

AUTHOR

- 2011 - All time top blog contributor to Planet MySQL
- 2010 - Published Author of Expert PHP & MySQL
- 2010 - Oracle ACE Director (first in MySQL)
- 2009 - MySQL community member of the year
- 22 years of RDBMS experience, 12 years with MySQL
  - MySQL Inc (2006-2008), Oracle Corp (96-99)
- Provide independent consulting/ Available NOW

MySQL Configuration

BEST PRACTICE

- Server SQL Mode

"Modes define what SQL syntax MySQL should support and what kind of data validation checks it should perform."

http://dev.mysql.com/doc/refman/5.1/en/server-sql-mode.html

SQL MODE

```
sql_mode = "STRICT_ALL_TABLES,
  NO_ZERO_DATE,
  NO_ZERO_IN_DATE,
  NO_ENGINE_SUBSTITUTION";
```

Minimum Recommended Configuration

```
mysql> SET GLOBAL
sql_mode="STRICT_ALL_TABLES,NO_ZERO_DATE,NO_ZERO_IN_DATE,NO_ENGINE_SUBSTITUTION";
```

```
# my.cnf
sql_mode="STRICT_ALL_TABLES,NO_ZERO_DATE,NO_ZERO_IN_DATE,NO_ENGINE_SUBSTITUTION"
```


SQL MODE

EXAMPLE

```
mysql> INSERT INTO orders (qty) VALUES (-1);
ERROR 1264 (22003): Out of range value for column 'i' at row 1
mysql> INSERT INTO orders (qty) VALUES (9000);
ERROR 1264 (22003): Out of range value for column 'i' at row 1
```

```
mysql> INSERT INTO orders (qty) VALUES (-1);
mysql> INSERT INTO orders (qty) VALUES (9000);
mysql> SELECT * FROM orders;
+-----+
| qty |
+-----+
|   0 |
| 255 |
+-----+
```

column is a 1 byte unsigned integer data type - TINYINT

SQL MODE

EXAMPLE

```
mysql> INSERT INTO orders (d) VALUES ('2010-02-31');
ERROR 1292 (22007): Incorrect date value: '2010-02-31' for
column 'd' at row 1
```

```
mysql> INSERT INTO orders (d) VALUES ('2010-02-31');
mysql> SELECT * FROM orders;
+------------+
| d          |
+------------+
| 0000-00-00 |
+------------+
```


STORAGE ENGINE

```
storage_engine="InnoDB"
```

Recommended Configuration

```
mysql> SET GLOBAL storage_engine="innodb";
```

```
# my.cnf
default-storage-engine=innodb
```


STORAGE ENGINE

EXAMPLE

```
mysql> INSERT INTO orders (order_id) VALUES (1),(2);
mysql> INSERT INTO orders (order_id) VALUES (3),(2);
ERROR 1062 (23000): Duplicate entry '2' for key 'PRIMARY'
mysql> SELECT * FROM orders;
+---+
| i |
+---+
| 1 |
| 2 |
+---+
```

```
mysql> INSERT INTO orders (order_id) VALUES (1),(2);
mysql> INSERT INTO orders (order_id) VALUES (3),(2);
ERROR 1062 (23000): Duplicate entry '2' for key 'PRIMARY'
mysql> SELECT * FROM orders;
+---+
| i |
+---+
| 1 |
| 2 |
| 3 |
+---+
```

Default MyISAM storage engine is non-transactional

STORAGE ENGINE

- Use a transactional storage engine
  - e.g. InnoDB
  - Other options exist
- MySQL default is MyISAM (*)
  - non-transactional
  - table level locking
  - No auto recovery

Changing to InnoDB in MySQL 5.5

Security

BASIC SECURITY

- Default is woeful
- Minimum
  - `$ mysql_secure_installation`
- Recommended
  - Operating System
  - Permissions & Privileges

OS SECURITY

- Defaults are not secure
- Never run as 'root' user
- Separate Data/Binary Logs/Logs/ Configuration/Backups
  - Individual directory permissions

- Minimize security risk
- Better auditability

INSTALLATION

- Best Practice

Single partition per MySQL Instance

```
/mysql
   /etc
   /data
   /binlog
   /log
   /mysql-version
```

Global files as symlinks from partition

```
/etc/my.cnf
/etc/profile.d/mysql.sh
/etc/init.d/mysqld
```

OS DATA SECURITY

- Software installed by root
  - Separate MySQL permissions for directories

```
$ chown -R root:root /mysql
$ chown -R mysql:mysql /mysql/{data,log,binlog,etc}
$ chmod 700 /mysql/{data,binlog}
$ chmod 750 /mysql/{etc,log}
```


APP USER SECURITY

- Username
- Password
- Host (source of username)

Best Practice

```
CREATE USER appuser@localhost IDENTIFIED BY 'sakila';
GRANT SELECT,INSERT,UPDATE,DELETE
      ON schema.* TO appuser@localhost;
```

Normal Practice

```
CREATE USER superman@'%';
GRANT ALL ON *.* TO superman@'%';
```


GRANT ALL

- GRANT ALL ON *.* TO user@'%'
  - *.* gives you access to all tables in all schemas
  - @'%' give you access from any external location
  - ALL gives you
    - ALTER, ALTER ROUTINE, CREATE, CREATE ROUTINE, CREATE TEMPORARY TABLES, CREATE USER, CREATE VIEW, DELETE, DROP, EVENT, EXECUTE, FILE, INDEX, INSERT, LOCK TABLES, PROCESS, REFERENCES, RELOAD, REPLICATION CLIENT, REPLICATION SLAVE, SELECT, SHOW DATABASES, SHOW VIEW, SHUTDOWN, **SUPER**, TRIGGER, UPDATE, USAGE

See MySQL Idiosyncrasies that BITE presentation -- http://rb42.com/idiosyncrasies

WHY SUPER IS BAD

- SUPER
  - Bypasses read_only
  - Bypasses init_connect
  - Can Disable binary logging
  - Change configuration dynamically
  - No reserved connection
  - ....

READ_ONLY

```
$ mysql -uappuser -psakila db
mysql> insert into test1(id) values(1);
ERROR 1290 (HY000): The MySQL server is running with the
--read-only option so it cannot execute this statement
```

```
$ mysql -usuperman db
mysql> insert into test1(id) values(1);
Query OK, 1 row affected (0.01 sec)
```

APP USER SECURITY

- Track Data Security
- Separation of responsibilities

- Application Viewer (Read Only Access)
  - SELECT
- Application User (Read/Write Access)
  - INSERT, UPDATE, DELETE, SELECT
  - CREATE TEMPORARY TABLE (if necessary)
- Application DBA (Schema Access Only)
  - CREATE, DROP, SELECT, INSERT, UPDATE, DELETE
  - CREATE ROUTINE (if necessary)

SQL

SQL IDEALS

- Comment your SQL
- Format your SQL
- Future proof your SQL
- Log your SQL
- Analyze your SQL
- Always use transactions

SQL COMMENTING

- Identify queries by code function
- Identify OLTP / Batch / Cache queries

DBA/SA can identify code function and purpose quickly

```
SELECT /* XXX123 */ ...
UPDATE /* YYY999 */ ...
SELECT /* Batch */...
```


SQL COMMENTING

EXAMPLE

```
  26 Query SELECT /* 5m cache */ .....
  26 Query SELECT /* ViewPost */ t.*, tt.*, tr.object_id FROM
wp_terms AS t INNER JOIN wp_term_taxonomy AS tt ON tt.term_id =
t.term_id INNER JOIN wp_term_relationships AS tr ON
tr.term_taxonomy_id = tt.term_taxonomy_id WHERE tt.taxonomy IN
('category', 'post_tag') AND tr.object_id IN (2849, 2842, 2836,
2824, 2812, 2680, 2813, 2800, 2770, 2784) ORDER BY t.name ASC
  26 Query SELECT /* batch */ key, value FROM usermeta WHERE user_id
= 2
```


SQL FORMATTING

- Create single line queries
  - Don't embed newlines
- Enables per line analysis by CLI tools

DBA/SA can use simple CLI tools including grep,awk,cut etc for SQL analysis

SQL FORMATTING

EXAMPLE

```
  26 Query SELECT FOUND_ROWS()
  26 Query SELECT post_id,start,end,allday,rpt,IF(end>='2010-06-04
00:00:00',1,0) AS active
     FROM wp_ec3_schedule
     WHERE post_id IN (2849,2842,2836,2824,2812,2680,2770,2784)
     ORDER BY start
  26 Query SELECT * FROM wp_users WHERE user_login = 'ronald'
  26 Query SELECT t.*, tt.*, tr.object_id FROM wp_terms AS t INNER
JOIN wp_term_taxonomy AS tt ON tt.term_id = t.term_id INNER JOIN
wp_term_relationships AS tr ON tr.term_taxonomy_id =
tt.term_taxonomy_id WHERE tt.taxonomy IN ('category', 'post_tag')
AND tr.object_id IN (2849, 2842, 2836, 2824, 2812, 2680, 2813, 2800,
2770, 2784) ORDER BY t.name ASC
  26 Query SELECT meta_key, meta_value FROM wp_usermeta WHERE
user_id = 2
```

SQL FORMATTING

EXAMPLE

```
$ cut -c12-80 general.log

SELECT FOUND_ROWS()
SELECT post_id,start,end,allday,rpt,IF(end>='2010-06-04',1,0
p_ec3_schedule
post_id IN (2849,2842,2836,2824,2812,2680,2770,2784)
BY start
SELECT * FROM wp_users WHERE user_login = 'ronald'
SELECT t.*, tt.*, tr.object_id FROM wp_terms AS t INNER JOIN ...
SELECT meta_key, meta_value FROM wp_usermeta WHERE user_id = 2
```


FUTURE PROOFING

- Specify INSERT column names
  - INSERT INTO table(a,b,c) VALUES(...)

```
mysql> INSERT INTO example VALUES (10,10,'A');

mysql> ALTER TABLE example ADD d DATE;

mysql> INSERT INTO example VALUES (10,10,'A');
ERROR 1136 (21S01): Column count doesn't match
value count at row 1
```

Reduce likelihood of runtime errors when structural changes to objects

FUTURE PROOFING

- Always use column aliases in joins
  - SELECT **a.**col1, **b.**col2 ...

```
mysql> SELECT id, name, val
     > FROM parent p, child c
     > WHERE p.id = c.parent_id;

mysql> alter table child add name varchar(10);

mysql> SELECT id, name, val FROM parent p, child
c WHERE p.id = c.parent_id;
ERROR 1052 (23000): Column 'name' in field list
is ambiguous
```


FUTURE PROOFING

- SELECT * is generally bad
  - What columns are actually used in code
  - TEXT/BLOB can cause extra disk I/O
  - New columns can change performance

FANCY CONSTRUCTS

- DELAYED
- IGNORE
- LOW PRIORITY
- REPLACE

- Changes ACID and statement precedence
- May have additional performance overhead

USE DETERMINISTIC

- Use '2010-06-21' instead of CURDATE()
  - Same for NOW()
- Don't Use ORDER BY RAND()
- Leverage Query Cache (if enabled)

SQL Analysis

GENERAL QUERY LOG

- Logs all SQL Statements
- Turn on for all development environments
- Aggregate and email results to developer
- Works best in single user environment

- Developers are seeing SQL in operation
- Enables access to SQL to analyze

GENERAL QUERY LOG

WARNING
Never enable in production

```
# my.cnf
general_log = ON
general_log_file = /mysql/log/general.log
log_output = FILE,TABLE
```

```
mysql> SET GLOBAL general_log=OFF;
mysql> SET GLOBAL general_log=ON;
mysql> SELECT event_time,argument FROM mysql.general_log;
+---------------------+---------------------------------------------------+
| event_time          | argument                                          |
+---------------------+---------------------------------------------------+
| 2010-10-11 21:48:05 | select * from test1                               |
| 2010-10-11 21:48:30 | SELECT event_time,argument FROM mysql.general_log |
+---------------------+---------------------------------------------------+
```

http://dev.mysql.com/doc/refman/5.1/en/query_log.html

GENERAL QUERY LOG

- For single user development environment
  - In SQL Session

```
mysql> SELECT 'Function X Start';
```

  - In Application
    - Run Function/Process
  - In SQL Session

```
mysql> SELECT 'Function X End';
```


GENERAL QUERY LOG

```
$ tail /mysql/log/general.log

101011 21:58:00     3 Query  SELECT 'Function X Start'
101011 21:58:09     2 Query  SELECT * from sample_int
101011 21:58:41     2 Query  SELECT * from example
101011 21:58:47     3 Query  SELECT 'Function X End'
```

http://dev.mysql.com/doc/refman/5.1/en/server-sql-mode.html

SQL ANALYSIS

- Bulk trending analysis
  - Volume of SQL statements
- Query Execution Plan (QEP)
  - Online v Batch/Cache SQL via commenting

- Identify bottlenecks ASAP without load
- Iterative Design feedback

Know your SQL

COMMON SQL ERRORS

- Remove redundant SQL
  - Use general query log
  - You may be surprised!

REPEATING QUERIES

The WRONG way

- SQL for one page load
  - 8 unwanted full table scans *BUG*
  - Removed gave 20x faster page load

```
5 Query    SELECT *  FROM `artist`
5 Query    SELECT *  FROM `artist`
5 Query    SELECT *  FROM `artist`
5 Query    SELECT *  FROM `artist`
5 Query    SELECT *  FROM `artist`
5 Query    SELECT *  FROM `artist` WHERE (ArtistID = 196 )
5 Query    SELECT *  FROM `artist` WHERE (ArtistID = 2188 )
5 Query    SELECT *  FROM `artist`
5 Query    SELECT * FROM `artist`
5 Query    SELECT *  FROM `artist`
```

Not cached (too big)
If cached, two queries

RAT V CAT

Row (RAT) Processing

```
SELECT * FROM activities_theme WHERE theme_parent_id=0
SELECT * FROM activities_theme WHERE theme_parent_id=1
SELECT * FROM activities_theme WHERE theme_parent_id=2
SELECT * FROM activities_theme WHERE theme_parent_id=11
SELECT * FROM activities_theme WHERE theme_parent_id=16
```

Chunk (CAT) Processing

```
SELECT *
FROM   activities_theme
WHERE  theme_parent_id in  (0,1,2,11,16)
```

ZERO RESULT QUERY

- The following SQL executed 6,000 times in 5 minute analysis period

The WRONG way

```
SELECT pages_id, pages_livestats_code, pages_title,
       pages_parent, pages_exhibid, pages_theme,
       pages_accession_num
FROM pages WHERE pages_id = 0
```

- 0 is an invalid pages_id

In a highly tuned system the greatest time in a query is network overhead

CODING ERRORS

- Remove duplicate code
- Affects Query Cache (if applicable)

- Less code to maintain
- Remove chance of human errors

DB CONNECTION

The WRONG way

```
$ cd public_html
$ grep "mysql*_connect" * */* */*/*
db-disp.php:$cid = mysql_connect("localhost", "museum", "******") or die ('I
cannot connect to the database because: ' . mysql_error());
test.php:$cid = mysql_connect("localhost", "museum", "******");
PMCollection/connection.php:  $dbcnx = mysql_connect("$sqlhost", "$sqluser",
"$sqlpass");
PMCollection/connection_live.php:  $dbcnx = mysql_connect("$sqlhost",
"$sqluser", "$sqlpass");
PMCollection/connection_local.php:  $dbcnx = mysql_connect("$sqlhost",
"$sqluser", "$sqlpass");
PMEcards/connection.php:  $dbcnx = mysql_connect("$sqlhost", "$sqluser",
"$sqlpass");
core/connection.php:      $dbcnx = mysql_connect("$sqlhost", "$sqluser",
"$sqlpass");
discussion_admin/db_fns.php:   $cid = mysql_connect("localhost", "museum",
"******");
discussion_admin/header.php:// $cid = mysql_connect("localhost", "museum",
"******");
discussion_admin/inc_title.php:    //$cid = mysql_connect("localhost",
"museum", "******");
discussion_admin/stats.php: //$cid = mysql_connect("localhost", "museum",
```

DB CONNECTION

The RIGHT way

```
class database {

  function getConnection($type, $message) {

    try {
      $con = mysqli_connect($cp->host,$cp->user,$cp->passwd,$cp->database);
      if (!$con) {
       $message = new message ("fatal", "Unable to obtain a '$type' ...
       return;
      }
      mysqli_query($con, "SET NAMES 'utf8'");
    } catch (Exception $e) {
       $message = new message ("fatal", "Unable to obtain a '$type' ...
       debug($e->getMessage());
    }

    return $con;
  }
```


DB CONNECTION

- Open and close database connections only when necessary

- Reduce unnecessary database load
- Increases page serve volume
- Increases true DB throughput

DB CONNECTION

The WRONG way

```
$ cat  header.php
...
$con = getConnection();
...

if($this_user->user_row["status"]!='ws' &&
    in_array($this_page->getValue(),$page)){
    header("Location: /permission.php");
    exit();
}
...
if () {
    header("Location: abc.php");
    exit();
}
...
if () {
    header("Location: xyz.php");
    exit();
}
...
```


DB CONNECTION

- Write Connection
- Read Connection
  - Connectors support for assignment

Conclusion

CONCLUSION

- Identify poor development practices
- Recommended best practices
- MySQL development tips and tricks
- Standard & expected RDBMS practices

See Part 2 for more tips