MySQL Best Practices

for DBAs and Developers

Volume I

OTN LAD Tour
South America
2010.10

Ronald Bradford
http://ronaldbradford.com

ORACLE TECHNOLOGY NETWORK

Oracle ACE

UYOUG

LAOUC

Agenda

- Essential MySQL configuration
- Improving your SQL
- MySQL user security
- Schema optimizations
- Instrumentation
- Monitoring

MySQL Configuration

MySQL Best Practices for DBAs and Developers - 2010.10

- 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

Configuration Best Practices

Minimum Recommended Configuration

Configuration Best Practices

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

Minimum Recommended Configuration

Configuration Best Practices

```
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 Examples

SQL Mode Examples

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

SQL Mode Examples

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

```
mysql> INSERT INTO sample_data (i) VALUES (-1), (9000);
mysql> SELECT * FROM sample_data;
+-----+
| i   |
+-----+
|   0 |
| 255 |
+-----+
```

Column is a 1 byte unsigned integer data type - TINYINT

SQL Mode Examples

SQL Mode Examples

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

SQL Mode Examples

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

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

Configuration Best Practices

Recommended Configuration

Configuration Best Practices

```
storage_engine="InnoDB"
```

Recommended Configuration

Configuration Best Practices

```
storage_engine="InnoDB"
```

Recommended Configuration

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

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

Configuration Best Practices

Configuration Best Practices

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

Configuration Best Practices

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

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

Default storage engine is non-transactional

Configuration Best Practices

- 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

Configuration Best Practices

- MySQL Idiosyncrasies that BITE presentation
  - Data Integrity
  - Storage Engine defaults
  - Referential Integrity

http://rb42.com/idiosyncrasies

Security

MySQL Physical Security

MySQL Physical Security

- Default is woeful

MySQL Physical Security

- Default is woeful
- Minimum

MySQL Physical Security

- Default is woeful
- Minimum
  - `$ mysql secure installation`

MySQL Physical Security

- Default is woeful
- Minimum
  - `$ mysql secure installation`
- Recommended

MySQL Physical Security

- Default is woeful
- Minimum
  - `$ mysql_secure_installation`
- Recommended
  - Operating System

MySQL Physical Security

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

Operating System 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

MySQL Installation Best Practice

- Best Practice

```
/mysql                          /etc/my.cnf
    /etc                        /etc/profile.d/mysql.sh
    /data                       /etc/init.d/mysqld
    /binlog
    /log
    /mysql-version
```

MySQL Installation Security

- Software installed by root
  - Separate MySQL permissions

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

Application User Security

Best Practice

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

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@'%';
```

Why not use GRANT ALL

Why not use GRANT ALL

- `GRANT ALL ON *.* TO user@'%'`

Why not use GRANT ALL

- `GRANT ALL ON *.* TO user@'%'`
  - *.* gives you access to all tables in all schemas

Why not use GRANT ALL

- `GRANT ALL ON *.* TO user@'%'`
  - *.* gives you access to all tables in all schemas
  - @'%' give you access from any external location

Why not use 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

Why not use 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

Application User Security Best Practices

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

- Track Data Security
- Separation of responsibilities

SQL

MySQL Best Practices for DBAs and Developers - 2010.10

Development Best Practices

- 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

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

• DBA/SA can identify code function and purpose quickly

SQL Commenting (2)

```
   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 (2)

```
   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
```

Future proofing your SQL

- Specify INSERT columns
  - `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 your SQL

- 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 your SQL

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

Avoid Fancy Constructs

- DELAYED
- IGNORE
- LOW PRIORITY
- REPLACE

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

Use Deterministic Functions

- Use '2010-06-21' instead of CURDATE()
  - Same for NOW()
- Don't Use ORDER BY RAND()

- Leverage Query Cache (if enabled)

•Leverages database caching when enabled
• Allows testing via parameterization

SQL Analysis

MySQL Best Practices for DBAs and Developers - 2010.10

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

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

Leveraging 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 Output

```
$ 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

MySQL Best Practices for DBAs and Developers - 2010.10

Common SQL Errors

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

Common Coding Errors - 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`
```

Common Coding Errors - 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

Common Coding Errors - Row at a time (RAT) Processing

The WRONG way

```
SELECT option_name, option_value FROM wp_options WHERE autoload = 'yes'
SELECT option_value FROM wp_options WHERE option_name = 'aiosp_title_format' LIMIT 1
SELECT option_value FROM wp_options WHERE option_name = 'ec3_show_only_even' LIMIT 1
SELECT option_value FROM wp_options WHERE option_name = 'ec3_num_months' LIMIT 1
SELECT option_value FROM wp_options WHERE option_name = 'ec3_day_length' LIMIT 1
SELECT option_value FROM wp_options WHERE option_name = 'ec3_hide_event_box' LIMIT 1
SELECT option_value FROM wp_options WHERE option_name = 'ec3_advanced' LIMIT 1
SELECT option_value FROM wp_options WHERE option_name = 'ec3_navigation' LIMIT 1
SELECT option_value FROM wp_options WHERE option_name = 'ec3_disable_popups' LIMIT 1
SELECT option_value FROM wp_options WHERE option_name = 'sidebars_widgets' LIMIT 1
```

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)
```

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)
```

Common Coding Errors - Boundary Conditions

- 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

Common Coding Errors

- Remove duplicate code

- Less code to maintain
- Remove chance of human errors

Defining your database 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",
```

Database connection example

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;
  }
```

Database Connections

- Open and close database connections only when necessary

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

Database Connections Initialization

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();
}
...
```

Schema Design

Design Best Practices

- Optimal Data Types
  - Saving Disk Space
- Naming Standards

About MySQL Data Types

- Numeric Data Types
  - Oracle has 1
  - MySQL has 9

```
TINYINT,SMALLINT,MEDIUMINT,INT,
BIGINT,FLOAT,DOUBLE,DECIMAL,BIT
```

Optimal Auto Increment Primary Key

- Don't use `BIGINT AUTO_INCREMENT`
- Use `INT UNSIGNED AUTO_INCREMENT`
- `BIGINT` is 8 bytes
- `INT` is 4 Bytes
- `INT UNSIGNED` stores 4.3 billion values

- Can reduce index space by 50+%
- Better memory usage, less I/O

Horror Stories

INT(1) is not what it looks like

- INT(1)
  - This is not 1 byte, it's 4 bytes
  - (1) is only for client display only
- Client with 10+ flags using INT(1)
- 40 bytes reduced to 10 bytes
  - or 2 bytes using bit operators

Dates

- MySQL supports
  - `DATE`, `DATETIME`, `TIMESTAMP`, `YEAR`
- `TIMESTAMP` for Epoch values
  - `TIMESTAMP` is 4 bytes
  - `DATETIME` is 8 bytes
  - Supports `DEFAULT CURRENT_TIMESTAMP`
- Neither store milliseconds

The beauty of ENUM

- Values Check Constraint
- Ideal for static codes
- Compact - i.e. 1 byte for 'True', 'False'
- Human readable values
- 5.1 Non blocking ALTER

TRUE/FALSE Datatype Examples

```
CREATE TABLE enums (
  flag1  CHAR(1) NULL COMMENT 'T or F, Y or N',
  flag2  TINYINT NULL COMMENT '0 or 1',
  flag3  BIT  NULL COMMENT 'True or False',
  flag4  ENUM ('True','False') NULL,
  flag5  VARCHAR(5) NULL
);
INSERT INTO enums(flag4, flag1, flag2)
VALUES ('True', 'T', 1), ('False', 'F', 0);
SELECT flag1, flag2, flag4 FROM enums;
+-------+-------+-------+
| flag1 | flag2 | flag4 |
+-------+-------+-------+
| T     |     1 | True  |
| F     |     0 | False |
+-------+-------+-------+
```

Schema Management

- Always have current schema.sql
- Use Patch/Revert SQL for upgrades
- See http://schemasync.org

- Reproducibility
- Upgrade/Downgrade path

Testing

Testing is not about what works; testing is about breaking your software

Resilience

- What is your backup strategy?
- What is your recovery strategy?
- How long does it take?
- Have you actually tested it end to end?

Take the survey

http://ronaldbradford.com/blog/checked-your-mysql-recovery-process-recently-2010-02-15/

Instrumentation

Application Instrumentation

- Creating one primary abstract DB call
- Enable logging of ALL SQL statements
- Enable embedded HTML output
- Total Execution Time/Count
- Individual SQL Execution Time/SQL

- Enable runtime analysis via browser
- No additional tools needed to gather

Memory Usage

5641240

17 queries logged

1. `SELECT * FROM j15svn_session WHERE session_id = 'ffb66b9026667c28f956dda21870ce00'`
2. `DELETE FROM j15svn_session WHERE ( time < 1186536522 )`
3. `SELECT * FROM j15svn_session WHERE session_id = 'ffb66b9026667c28f956dda21870ce00'`
4. `INSERT INTO j15svn_session ( `session_id`,`time`,`client_id` ) VALUES ( 'ffb66b9026667c28f956dda21870ce00','1186590522',`
5. `SELECT id, name, folder, element, published, params FROM j15svn_plugins WHERE published >= 1 AND access <= 0 ORDER BY or`
6. `SELECT m.*, c.`option` AS component FROM j15svn_menu AS m LEFT JOIN j15svn_components AS c ON m.componentid = c.id WHERE`
7. `SELECT * FROM j15svn_components WHERE parent = 0`
8. `SELECT template, menuid FROM j15svn_templates_menu WHERE client_id = 0`
9. `SELECT a.id, a.title, a.title_alias, a.introtext, a.sectionid, a.state, a.catid, a.created, a.created_by, a.created_by_a`
   `a.checked_out_time, a.publish_up, a.publish_down, a.images, a.attribs, a.urls, a.ordering, a.metakey, a.metadesc, a.acces`
   `a.alias) ELSE a.id END AS slug, CHAR_LENGTH( a.`fulltext` ) AS readmore, u.name AS author, u.usertype, g.name AS groups,`
   `j15svn_content_frontpage AS f ON f.content_id = a.id LEFT JOIN j15svn_categories AS cc ON cc.id = a.catid LEFT JOIN j15s`
   `ON a.access = g.id WHERE 1 AND a.access <= 0 AND a.state = 1 AND ( publish_up = '0000-00-00 00:00:00' OR publish_up <= '`
   `OR publish_down >= '2007-08-08 16:28:42' ) ORDER BY  f.ordering LIMIT 0, 9`
10. `SELECT a.id, a.title, a.title_alias, a.introtext, a.sectionid, a.state, a.catid, a.created, a.created_by, a.created_by_a`
    `a.checked_out_time, a.publish_up, a.publish_down, a.images, a.attribs, a.urls, a.ordering, a.metakey, a.metadesc, a.acces`
    `a.alias) ELSE a.id END AS slug, CHAR_LENGTH( a.`fulltext` ) AS readmore, u.name AS author, u.usertype, g.name AS groups,`
    `j15svn_content_frontpage AS f ON f.content_id = a.id LEFT JOIN j15svn_categories AS cc ON cc.id = a.catid LEFT JOIN j15s`
    `ON a.access = g.id WHERE 1 AND a.access <= 0 AND a.state = 1 AND ( publish_up = '0000-00-00 00:00:00' OR publish_up <= '`
    `OR publish_down >= '2007-08-08 16:28:42' ) ORDER BY  f.ordering`
11. `SELECT id, title, module, position, content, showtitle, control, params FROM j15svn_modules AS m LEFT JOIN j15svn_modules`
    `m.access <= 0 AND m.client_id = 0 AND ( mm.menuid = 1 OR mm.menuid = 0 ) ORDER BY position, ordering`
12. `SELECT id, title FROM j15svn_polls WHERE id = 14 AND published = 1`
13. `SELECT id, text FROM j15svn_poll_data WHERE pollid = 14 AND text <> "" ORDER BY id`
14. `SELECT guest, usertype, client_id FROM j15svn_session WHERE client_id = 0`
15. `SELECT a.*, CASE WHEN CHAR_LENGTH(a.alias) THEN CONCAT_WS(":", a.id, a.alias) ELSE a.id END AS slug, CASE WHEN CHAR_LENG`
    `AS catslug FROM j15svn_content AS a LEFT JOIN j15svn_content_frontpage AS f ON f.content_id = a.id INNER JOIN j15svn_cat`
    `ON s.id = a.sectionid WHERE ( a.state = 1 AND s.id > 0 ) AND ( a.publish_up = "0000-00-00 00:00:00" OR a.publish_up <= "`
    `00:00:00" OR a.publish_down >= "2007-08-08 16:28:42" ) AND a.access <= 0 AND cc.access <= 0 AND s.access <= 0 AND s.publ`
16. `SELECT a.*,  CASE WHEN CHAR_LENGTH(a.alias) THEN CONCAT_WS(":", a.id, a.alias) ELSE a.id END AS slug, CASE WHEN CHAR_LEN`
    `AS catslug FROM j15svn_content AS a INNER JOIN j15svn_categories AS cc O`
    `"0000-00-00 00:00:00" OR a.publish_up <= "2007-08-08 16:28:42" ) AND (`
    `AND cc.access <= 0 AND s.access <= 0 AND s.published = 1 AND cc.publish`
17. `SELECT a.*, CASE WHEN CHAR_LENGTH(a.alias) THEN CONCAT_WS(":", a.id,`
    `catslug FROM j15svn_content AS a INNER JOIN j15svn_categories AS cc ON c`
    `cc.access <= 0 AND s.access <= 0 AND (a.publish_up = "0000-00-00 00:00:00" OR a.publish_up <= "2007-08-08 16:28:42" )`
    `"2007-08-08 16:28:42" ) AND cc.id = 3 AND cc.section = s.id AND cc.publishe`

•SQL Statements Example

Language Files Loaded

111 ms	111 ms	1	/home/sitedirs/example/site/account/column_right.cfm
109 ms	109 ms	1	/home/example/public_html/includes/marketplace.cfm
15 ms	15 ms	1	/home/sitedirs/example/site/layout/leaderboard.cfm
14 ms	14 ms	1	/home/example/public_html/includes/layout/leaderboard.cfm
13 ms	13 ms	1	/home/example2/public_html/includes/scripts/ad.cfm
13 ms	13 ms	1	/home/sitedirs/example/site/account/column_body_left.cfm
12 ms	12 ms	1	/home/example/public_html/account/includes/main.cfm
8 ms	8 ms	1	/home/sitedirs/example/site/layout/searchbar.cfm
4 ms	4 ms	1	/home/example/public_html/Application.cfm
1 ms	1 ms	1	/home/example/public_html/OnRequestEnd.cfm
1 ms	1 ms	1	/home/sitedirs/example/site/layout/searchbar_form.cfm
0 ms	0 ms	1	/home/example/public_html/account/includes/newsletters.cfm
0 ms	0 ms	1	/home/example/public_html/includes/google_analytics.cfm
0 ms	0 ms	1	/home/sitedirs/example/site/Application.cfm
0 ms	0 ms	1	/home/sitedirs/example/site/includes/create_account.cfm
0 ms	0 ms	1	/home/sitedirs/example/site/includes/navmenu.cfm
0 ms	0 ms	1	/home/sitedirs/example/site/layout/footer.cfm
2 ms			*STARTUP, PARSING, COMPILING, LOADING, & SHUTDOWN*
162 ms			*TOTAL EXECUTION TIME*

red = over 250 ms average execution time

•SQL Statements Example

SQL Queries `table1` (Datasource=users, Time=0ms, Records=1, Cached Query) in /home/example/public_html/Application.cfm @ 15:42:56.056

```
  select * from users.table1 where id='41'
```

check_rt (Datasource=users, Time=2ms, Records=1) in /home/sitedirs/example/site/sections/section_body.cfm @ 15:42:56.056

```
        select rivertown from table1
        where id='41'
```

ad_juggler (Datasource=users, Time=11ms, Records=1) in /home/example2/public_html/includes/scripts/ad.cfm @ 15:42:56.056

```
    select *
    from users.table1 where id='41'
```

searchbar_user (Datasource=users, Time=1ms, Records=1) in /home/sitedirs/example/site/layout/searchbar.cfm @ 15:42:56.056

```
          select first,last,email from users where users.id='178233'
```

Application Instrumentation Benefits

- End to End Timing
- Component Timing
  - (i.e. a series of SQL)
- Observe as desired
- Intelligent Activation
  - e.g. Page Load exceeds x ms

Monitoring

If you don't have monitoring in place, make it your top priority

Monitoring

- Monitoring
- Alerting
- Dashboard
- Public Status

Successful Scalability

http://ronaldbradford.com/blog/successful-mysql-scalability-presentation-2010-09-17/

Don't change a setting without evidence to prove/ disprove the

Bad MySQL Configuration

- [sort|join|read|read_rnd] _buffer_size
  - Defaults are 128K - 256K
  - Settings of 1M, 2M, 16M, 128M
- Pre allocated per thread buffer
  - Larger buffer is slower to create > 256K
  - Wastes valuable process memory

Where is the bottleneck?

- Is the database the problem?
- Front End Performance
  - The website will always be too slow
  - Identify the true components
- End to End time
  - Database may be only a small portion

Conclusion

Objectives

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

Questions?

RonaldBradford.com

MySQL Best Practices for DBAs and Developers - 2010.10