EffectiveMySQL.com

Performance, Scalability & Business Continuity

Explaining the MySQL EXPLAIN

Ronald Bradford
http://ronaldbradford.com

RMOUG QEW
Denver, Colorado - 2012.05

AGENDA

- EXPLAIN syntax options
- How to read QEP
- QEP examples
- MySQL optimizer limitations

Slides at http://j.mp/EM-Explain

PURPOSE

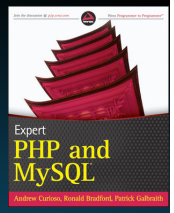
EXPLAIN is used for

- Determine Query Execution Plan (QEP)
- Understand MySQL optimizer
- Identify information needed for tuning
- Not providing tuning recommendations

ABOUT THE AUTHOR

RONALD BRADFORD

- 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 Corporation (1996-1999)
- Provide independent consulting - Available NOW



MYSQL QEP

- Cost Based Optimizer
- No pinning
- Few hints
- Calculated every SQL execution
  - Except Query Cache

SYNTAX

- `EXPLAIN SELECT ...`
  - `EXPLAIN PARTITIONS SELECT ...`
  - `EXPLAIN EXTENDED SELECT ...`
- Does not support UPDATE,DELETE

Now supported in MySQL 5.6

SUPPORT COMMANDS

- SHOW CREATE TABLE
- SHOW INDEXES
- SHOW TABLE STATUS
- INFORMATION_SCHEMA
- SHOW VARIABLES
- SHOW STATUS
- Optimizer Trace - Since 5.6.3

EXPLAIN USAGE

- Does not execute query

  BUT

- May execute portions
  - e.g. derived tables

WARNING !!!

EXPLAIN

EXAMPLE

```
mysql> EXPLAIN SELECT * FROM proposal WHERE post_id=16102176;
+----+-------------+----------+------+---------------+------+---------+------+--------+-------------+
| id | select_type | table    | type | possible_keys | key  | key_len | ref  | rows   | Extra       |
+----+-------------+----------+------+---------------+------+---------+------+--------+-------------+
|  1 | SIMPLE      | proposal | ALL  | NULL          | NULL | NULL    | NULL | 787126 | Using where |
+----+-------------+----------+------+---------------+------+---------+------+--------+-------------+
```

```
mysql> EXPLAIN SELECT * FROM inventory
    -> WHERE item_id = 16102176\G
*************************** 1. row ***********************
           id: 1
  select_type: SIMPLE
        table: inventory
         type: ref
possible_keys: item_id
          key: item_id
      key_len: 4
          ref: const
         rows: 1
        Extra:
```

MySQL Client SQL Statement Terminator
;
\G

ATTRIBUTES

- id
- select_type
- table
- type
- possible_keys
- key
- key_len
- ref
- rows
- Extra

```
mysql> EXPLAIN SELECT * FROM inventc
    -> WHERE item_id = 16102176\G
*************************** 1. row **
           id: 1
  select_type: SIMPLE
        table: inventory
         type: ref
possible_keys: item_id
          key: item_id
      key_len: 4
          ref: const
         rows: 1
        Extra:
```

ESSENTIAL EXAMPLE

- id
- select_type
- table
- type
- possible_keys
- **key**
- key_len
- ref
- **rows**
- Extra

```
mysql> EXPLAIN SELECT * FROM inventc
    -> WHERE item_id = 16102176\G
*************************** 1. row ***
           id: 1
  select_type: SIMPLE
        table: inventory
         type: ref
possible_keys: item_id
          key: item_id
      key_len: 4
          ref: const
         rows: 1
        Extra:
```


ESSENTIAL EXAMPLE

- id
- select_type
- table
- type
- possible_keys
- **key**
- key_len
- ref
- **rows**
- Extra

```
mysql> EXPLAIN SELECT * FROM inventc
    -> WHERE item_id = 16102176\G
*************************** 1. row ***
           id: 1
  select_type: SIMPLE
        table: inventory
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 787338
        Extra: Using where
```


ESSENTIAL EXAMPLE

- id
- select_type
- table
- type
- possible_keys
- **key**
- key_len
- ref
- **rows**
- Extra

```
mysql> EXPLAIN SELECT * FROM inventc
    -> WHERE item_id = 16102176\G
*************************** 1. row ***
           id: 1
  select_type: SIMPLE
        table: inventory
         type: ref
possible_keys: item_id
          key: item_id
      key_len: 4
          ref: const
         rows: 1
        Extra:
```


key

- Identify index to be used
- Generally only one per table (*)
- Associated attributes
  - possible_keys
  - key_len

```
           id: 1
  select_type: SIMPLE
        table: inventc
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 787338
        Extra: Using w
```

key

EXAMPLE

```
mysql> EXPLAIN SELECT * FROM inventory
    -> WHERE item_id = 16102176\G
*************************** 1. row ***********************
           id: 1
  select_type: SIMPLE
        table: inventory
         type: ref
possible_keys: item_id
          key: item_id
      key_len: 4
          ref: const
         rows: 1
        Extra:
```


key

MERGE INDEX EXAMPLE

```
mysql> EXPLAIN SELECT id FROM users
    -> WHERE first = 'west' OR last='west'\G
************************ 1. row **********************
           id: 1
  select_type: SIMPLE
        table: users
         type: index_merge
possible_keys: first,last
          key: first,last
      key_len: 22,22
          ref: NULL
         rows: 2
        Extra: Using union(first,last); Using where
```

```
CREATE TABLE `users` (
  `id` int(10) unsigned NOT NULL AU
  `first` varchar(20) NOT NULL,
  `last` varchar(20) NOT NULL,
  `username` varchar(20) NOT NULL,
  `last_login` timestamp NOT NULL D
  PRIMARY KEY (`id`),
  KEY `first` (`first`),
  KEY `last` (`last`),
  KEY `username` (`username`) ...
```


possible_keys

- Indexes the optimizer considered
  - Why was no index used?
  - Too many is not good
- Associated attributes
  - key

```
           id: 1
  select_type: SIMPLE
        table: inventc
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 787338
        Extra: Using w
```


possible_keys

EXAMPLE

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	c	const	PRIMARY	PRIMARY	4	const	1	Using filesort
1	SIMPLE	i	ALL	**customer_id**	NULL	NULL	NULL	7	Using where

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	c	const	PRIMARY	PRIMARY	4	const	1	
1	SIMPLE	i	**ref**	customer_id	**customer_id**	4	const	5	Using where

possible_keys
EXAMPLE

```
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: FES
         type: ref
possible_keys: FK_ASI,EmpID1,Idx_SID,SuleUnitsEmpSuled,Idx_SUnitID
          key: SuleUnitsEmpSuled
      key_len: 8
          ref: FSU.SUnitId,FS.SID
         rows: 26
        Extra:
```

```
..
  PRIMARY KEY (`ID`),
  KEY `FK_ASI` (`ASID`),
  KEY `EmpID1` (`EmpID`),
  KEY `Idx_Composite` (`ID`,`EmpID`,`SUnitID`,`SID`,`Sou
  KEY `Idx_SID` (`SID`),
  KEY `SuleUnitsEmpSuled` (`SUnitID`,`SID`),
  KEY `Idx_SUnitID` (`SUnitID`),
```


rows

- Estimated number of table rows (*)
- Associated attributes
  - key

```
           id: 1
  select_type: SIMPLE
        table: invento
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 787338
        Extra: Using w
```


key_len

- Amount of index used (*)
  - Multi column efficiency
- Associated attributes
  - Extra = Using Index

```
           id: 1
  select_type: SIMPLE
        table: invento
         type: ALL
possible_keys: inv_id
          key: inv_id
      key_len: 4
          ref: NULL
         rows: 1
        Extra: Using w
```


key_len
CALCULATIONS

- TINYINT - 1 byte
- SMALLINT - 2 bytes
- INT - 4 bytes
- BIGINT - 8 bytes
- DATE - 3 bytes
- TIMESTAMP - 4 bytes
- DATETIME - 8 bytes

- CHAR(n) - n bytes
- VARCHAR(n) - n bytes
- NULL + 1 byte
- VARCHAR + 2 bytes
- Character set x [1-3] bytes

key_len

EXAMPLE

```
mysql> EXPLAIN SELECT user_id,balance,created
    -> FROM accounts
    -> WHERE id = 42\G
*************************** 1. row ***********************
           id: 1
  select_type: SIMPLE
        table: accounts
         type: const
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 8
          ref: const
         rows: 1
        Extra:
```

8 Bytes

```
CREATE TABLE `accounts` (
  `id` BIGINT NOT NULL AUTO_INCREMEN
...
PRIMARY KEY (id)
...
```


key_len with varchar

EXAMPLE

```
mysql> EXPLAIN SELECT *
    -> FROM categories
    -> WHERE name LIKE 'NCAA%';
*************************** 1. row ***********************
           id: 1
  select_type: SIMPLE
        table: categories
         type: range
possible_keys: name
          key: name
      key_len: 32
          ref: NULL
         rows: 6
        Extra: Using where
```

30 Bytes + 2 Bytes

```
CREATE TABLE categories (
  id int NOT NULL AUTO_INCREMENT,
  name VARCHAR(30) NOT NULL,
...
  INDEX (name)
...
```


key_len with utf8

EXAMPLE

```
mysql> EXPLAIN SELECT *
    -> FROM categories
    -> WHERE name LIKE 'NCAA%';

*************************** 1. row ***********************
           id: 1
  select_type: SIMPLE
        table: categories
         type: range
possible_keys: name
          key: name
      key_len: 92
          ref: NULL
         rows: 6
        Extra: Using where
```

(30 * 3) Bytes + 2 Bytes

```
CREATE TABLE `categories` (
  `id` int(4) NOT NULL AUTO_INCREMEN
  `name` VARCHAR(30) NOT NULL,
...
  INDEX (name)
) ... DEFAULT CHARSET=utf8
```


key_len

EXAMPLE

(1) (3)

```
mysql> EXPLAIN SELECT ID, post_title FROM wp_posts WHERE post_type='post' and post_date > '2010-06-01';
+----+-------------+----------+------+-----------------+-----------------+---------+-------+------+-------------+
| id | select_type | table    | type | possible_keys   | key             | key_len | ref   | rows | Extra       |
+----+-------------+----------+------+-----------------+-----------------+---------+-------+------+-------------+
|  1 | SIMPLE      | wp_posts | ref  | type_status_date | type_status_date | 62      | const | 1132 | Using where |
+----+-------------+----------+------+-----------------+-----------------+---------+-------+------+-------------+
```

```
CREATE TABLE `wp_posts` (
...
  `post_date` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `post_status` varchar(20) NOT NULL DEFAULT 'publish',
  `post_type` varchar(20) NOT NULL DEFAULT 'post',
...
  PRIMARY KEY (`ID`),
  KEY `type_status_date` (`post_type`,`post_status`,`post_date`,`ID`),
) DEFAULT CHARSET=utf8
```

(1) (2) (3) (4)

62 + 62 + 8 + 8 Bytes

key_len

EXAMPLE

(1) (3)

```
mysql> EXPLAIN SELECT ID, post_title FROM wp_posts WHERE post_type='post' and post_date > '2010-06-01';
+----+-------------+----------+------+-----------------+-----------------+---------+-------+------+-------------+
| id | select_type | table    | type | possible_keys   | key             | key_len | ref   | rows | Extra       |
+----+-------------+----------+------+-----------------+-----------------+---------+-------+------+-------------+
|  1 | SIMPLE      | wp_posts | ref  | type_status_date | type_status_date | 62      | const | 1132 | Using where |
+----+-------------+----------+------+-----------------+-----------------+---------+-------+------+-------------+
```

(1) (2) (3)

```
mysql> EXPLAIN SELECT ID, post_title FROM wp_posts WHERE post_type='post' AND post_status='publish' AND post_date >
'2010-06-01';
+----+-------------+----------+------+-----------------+-----------------+---------+-------+------+-------------+
| id | select_type | table    | type | possible_keys   | key             | key_len | ref   | rows | Extra       |
+----+-------------+----------+------+-----------------+-----------------+---------+-------+------+-------------+
|  1 | SIMPLE      | wp_posts | range | type_status_date | type_status_date | 132     | NULL  |    1 | Using where |
+----+-------------+----------+------+-----------------+-----------------+---------+-------+------+-------------+
```

62+62+8 Bytes

```
CREATE TABLE `wp_posts` (
...
  `post_date` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `post_status` varchar(20) NOT NULL DEFAULT 'publish',
  `post_type` varchar(20) NOT NULL DEFAULT 'post',
...
  PRIMARY KEY (`ID`),
  KEY `type_status_date` (`post_type`,`post_status`,`post_date`,`ID`),
) DEFAULT CHARSET=utf8
```

(1) (2) (3) (4)

62 + 62 + 8 + 8 bytes

select_type

- SIMPLE
- PRIMARY
- SUBQUERY
- DERIVED
- UNION
- DEPENDENT UNION
- UNION RESULT
- UNCACHEABLE QUERY
- UNCACHEABLE UNION

```
           id: 1
  select_type: SIMPLE
        table: inventc
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 787338
        Extra: Using w
```


select_type

EXAMPLE

```
mysql> EXPLAIN SELECT MAX(id) FROM (SELECT id FROM users WHERE first = 'west') c;
+----+-------------+-----------+------+---------------+-------+---------+------+------+-------------+
| id | select_type | table     | type | possible_keys | key   | key_len | ref  | rows | Extra       |
+----+-------------+-----------+------+---------------+-------+---------+------+------+-------------+
|  1 | PRIMARY     | <derived2> | ALL  | NULL          | NULL  | NULL    | NULL |    2 |             |
|  2 | DERIVED     | users     | ref  | first         | first | 22      |      |    1 | Using where |
+----+-------------+-----------+------+---------------+-------+---------+------+------+-------------+

mysql> EXPLAIN SELECT p.* FROM parent p WHERE p.val LIKE 'a%'
    -> UNION
    -> SELECT p.* FROM parent p WHERE p.id > 5;
+----+-------------+-----------+------+---------------+------+---------+------+------+-------------+
| id | select_type | table     | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+----+-------------+-----------+------+---------------+------+---------+------+------+-------------+
|  1 | PRIMARY     | p         | range | val           | val  | 12      | NULL |    1 | Using where |
|  2 | UNION       | p         | ALL  | PRIMARY       | NULL | NULL    | NULL |    8 | Using where |
| NULL | UNION RESULT | <union1,2> | ALL  | NULL          | NULL | NULL    | NULL | NULL |             |
+----+-------------+-----------+------+---------------+------+---------+------+------+-------------+
```

2 represents id

1,2 represents id

select_type

EXAMPLE

3 ways to get same query results

```
mysql> EXPLAIN SELECT p.* FROM parent p WHERE p.id NOT IN (SELECT c.parent_id FROM child c);
+----+--------------------+-------+-----------------+---------------+----------+---------+------+------+-------------+
| id | select_type        | table | type            | possible_keys | key      | key_len | ref  | rows | Extra       |
+----+--------------------+-------+-----------------+---------------+----------+---------+------+------+-------------+
|  1 | PRIMARY            | p     | ALL             | NULL          | NULL     | NULL    | NULL |    7 | Using where |
|  2 | DEPENDENT SUBQUERY | c     | index_subquery  | parent_id     | parent_id | 4       | func |    2 | Using index |
+----+--------------------+-------+-----------------+---------------+----------+---------+------+------+-------------+

mysql> EXPLAIN SELECT p.* FROM parent p LEFT JOIN child c ON p.id = c.parent_id WHERE c.child_id IS NULL;
+----+-------------+-------+------+---------------+----------+---------+------+------+--------------------------+
| id | select_type | table | type | possible_keys | key      | key_len | ref  | rows | Extra                    |
+----+-------------+-------+------+---------------+----------+---------+------+------+--------------------------+
|  1 | SIMPLE      | p     | ALL  | NULL          | NULL     | NULL    | NULL |    7 |                          |
|  1 | SIMPLE      | c     | ref  | parent_id     | parent_id | 4       | p.id |    2 | Using where; Not exists |
+----+-------------+-------+------+---------------+----------+---------+------+------+--------------------------+

mysql> EXPLAIN SELECT p.* FROM parent p WHERE  NOT EXISTS (SELECT parent_id FROM child c WHERE c.parent_id = p.id);
+----+--------------------+-------+------+---------------+----------+---------+------+------+-------------+
| id | select_type        | table | type | possible_keys | key      | key_len | ref  | rows | Extra       |
+----+--------------------+-------+------+---------------+----------+---------+------+------+-------------+
|  1 | PRIMARY            | p     | ALL  | NULL          | NULL     | NULL    | NULL |    7 | Using where |
|  2 | DEPENDENT SUBQUERY | c     | ref  | parent_id     | parent_id | 4       | p.id |    2 | Using index |
+----+--------------------+-------+------+---------------+----------+---------+------+------+-------------+
```

Which query is best?

Extra

Most Common

- Using where
- Using temporary
- Using filesort
- Using index **GOOD**
- Using join buffer

```
           id: 1
  select_type: SIMPLE
        table: invento
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 787338
        Extra: Using w
```


Extra - Using temporary

- Internal Table (MEMORY based)
- Can have multiple per query
- Save to disk impact
  - TEXT/BLOB
  - Size > `min(max_heap_table_size, tmp_table_size)`

http://forge.mysql.com/wiki/Overview_of_query_execution_and_use_of_temp_tables

Extra - Using filesort

- ORDER BY
  - Can be CPU intensive
- Is order via DB necessary?
- Can you leverage an index?

Extra - Using filesort

EXAMPLE

```
EXPLAIN
SELECT   i.invoice_date, i.customer_id, i.invoice_total, c.company, c.state
FROM     invoice i INNER JOIN customer c USING (customer_id)
WHERE    i.customer_id = 42
ORDER BY i.invoice_date;
```

table	type	possible_keys	key	key_len	ref	rows	Extra
c	const	PRIMARY	PRIMARY	4	const	1	**Using filesort**
i	ref	customer_id	customer_id	4	const	5	Using where

```
CREATE TABLE invoice(
  invoice_id  INT UNSIGNED NOT NULL AUTO_INCREMENT,
  invoice_date DATE NOT NULL,
  customer_id  INT UNSIGNED NOT NULL,
  invoice_total DECIMAL(10,2) NOT NULL,
  PRIMARY KEY(invoice_id),
  KEY (customer_id)
) ENGINE=InnoDB;
```

Extra - Using filesort

EXAMPLE

```
EXPLAIN
SELECT   i.invoice_date, i.customer_id, i.invoice_total, c.company, c.state
FROM     invoice i INNER JOIN customer c USING (customer_id)
WHERE    i.customer_id = 42
ORDER BY i.invoice_date;
```

table	type	possible_keys	key	key_len	ref	rows	Extra
c	const	PRIMARY	PRIMARY	4	const	1	~~Using filesort~~
i	ref	customer_id	customer_id	4	const	5	Using where

```
CREATE TABLE invoice(
  invoice_id  INT UNSIGNED NOT NULL AUTO_INCREMENT,
  invoice_date DATE NOT NULL,
  customer_id  INT UNSIGNED NOT NULL,
  invoice_total DECIMAL(10,2) NOT NULL,
  PRIMARY KEY(invoice_id),
  KEY (customer_id, invoice_date)
) ENGINE=InnoDB;
```

Modify index, remove per query sorting

Extra - Using index

- Does not mean using index
- Means using ONLY THE INDEX
- Additional Presentation

Improving Performance with Better Indexes

Extra - Using index

EXAMPLE

Executed 25-30 thousand (30,000) queries per second

id	select_t	tab	key	key	id	select_t	tab	key	key_	Extra
1	PRIMARY	c	statu	1	1	PRIMARY	c	adver	4	Using where
1	PRIMARY	e	PRIMA	4	1	PRIMARY	e	**statu**	**1**	Using where; **Using index**
1	PRIMARY	g	campa	4	1	PRIMARY	g	campa	4	Using where
10	DEPENDEN	crb	id_ca	**4**	10	DEPENDEN	crb	id_ca	**66**	Using where
9	DEPENDEN	csb	pub_s	98	9	DEPENDEN	csb	pub_s	98	Using where
8	DEPENDEN	arb	id_ad	**4**	8	DEPENDEN	arb	id_ad	**26**	Using where
7	DEPENDEN	asb	pub_s	**34**	7	DEPENDEN	asb	**id_ad**	**40**	Using where; **Using index**
6	DEPENDEN	pm	id_adr	**4**	6	DEPENDEN	pm	id_adr	**12**	Using index
5	DEPENDEN	tgv	searc	**4**	5	DEPENDEN	tgv	searc	**10**	Using where; **Using index**
4	DEPENDEN	st	id_sc	4	4	DEPENDEN	st	id_sc	4	Using where; **Using index**
4	DEPENDEN	t	PRIMA	4	4	DEPENDEN	t	PRIMA	4	Using where
3	DEPENDEN	k2	keywo	302	3	DEPENDEN	k2	keywo	302	Using where; **Using index**
3	DEPENDEN	gk2	PRIMA	100	3	DEPENDEN	gk2	PRIMA	100	Using where
2	DEPENDEN	k1	keywo	302	2	DEPENDEN	k1	keywo	302	Using where; **Using index**
2	DEPENDEN	gk1	PRIMA	100	2	DEPENDEN	gk1	PRIMA	100	Using where

Before new indexes executed in 175ms

After new indexes executed in 5ms

Extra - Using join buffer

EXAMPLE

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	fs	ref	PRIMARY,sts	sts	768	const	21	Using where; Using temporary; U
1	SIMPLE	fsu	ref	PRIMARY,d,sid	sid	4	fs.sid	21	Using where
1	SIMPLE	fes	ref	sasi,eid,sid,sues,suid	sues	8	fsi.suid,fs.sid	26	
1	SIMPLE	ma	ALL	masi	**NULL**	NULL	NULL	200	Using where; **Using join buffer**
1	SIMPLE	fas	eq_ref	PRIMARY	PRIMARY	4	ma.faid	1	Using where; Using index
1	SIMPLE	las	eq_ref	PRIMARY,asai	PRIMARY	4	ma.laid	1	
1	SIMPLE	la	eq_ref	PRIMARY	PRIMARY	4	las.aid	1	
1	SIMPLE	fp	eq_ref	PRIMARY	PRIMARY	8	ses.eid	1	

No index can be satisfied for join condition.
i.e. full table scan

Extra cont.

Less common

- Impossible WHERE ...
- Distinct
- Not exists
- Select tables optimized away

Extra EXAMPLE

```
mysql> EXPLAIN SELECT COUNT(*) FROM (SELECT id FROM users WHERE first = 'west') c
*************************** 1. row ***************************
           id: 1
  select_type: PRIMARY
        table: NULL
         type: NULL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: NULL
        Extra: Select tables optimized away
*************************** 2. row ***************************
           id: 2
  select_type: DERIVED
        table: users
         type: ref
possible_keys: first
          key: first
      key_len: 22
          ref:
         rows: 1
```


Extra cont.

Merge Indexes

- Using sort_union(...)
- Using union(...)
- Using intersect(...)

SYNTAX

- **EXPLAIN** SELECT ...
  - **EXPLAIN PARTITIONS** SELECT ...
  - **EXPLAIN EXTENDED** SELECT ...

EXPLAIN PARTITIONS

EXAMPLE

```
   mysql> EXPLAIN PARTITIONS SELECT * from audit_log WHERE yr in
   (2011,2012)\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: audit_log
   partitions: p2,p3
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 2
        Extra: Using where
```


EXPLAIN EXTENDED

EXAMPLE

```
   mysql> EXPLAIN EXTENDED select t1.name from test1 t1 INNER JOIN test2 t2 USING(uid)\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: t1
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 1
     filtered: 100.00
        Extra:
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: t2
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 98
          ref: func
         rows: 1
     filtered: 100.00
        Extra: Using where; Using index
2 rows in set, 1 warning (0.00 sec)
```


EXPLAIN EXTENDED

EXAMPLE

```
mysql> EXPLAIN EXTENDED select t1.name from test1 t1 INNER JOIN
test2 t2 USING(uid)\G


mysql> SHOW WARNINGS\G
*************************** 1. row ***************************
  Level: Note
   Code: 1003
Message: select `book`.`t1`.`name` AS `name` from `book`.`test1`
`t1` join `book`.`test2` `t2` where (convert(`book`.`t1`.`uid` using
utf8) = `book`.`t2`.`uid`)
```


INDEX HINTS

- USE INDEX
- IGNORE INDEX
- FORCE INDEX

Can specify multiple indexes

- FOR
  - JOIN | ORDER BY | GROUP BY

SELECT HINTS

- STRAIGHT_JOIN
  - Defines order of tables in QEP

SELECT HINTS

- SQL_CACHE
- SQL_NO_CACHE
- SQL_CALC_FOUND_ROWS
- SQL_BIG_RESULT
- SQL_SMALL_RESULT
- SQL_BUFFER_RESULT
- HIGH_PRIORITY

SQL_CALC_FOUND_ROWS

EXAMPLE

```
SELECT id,username FROM users WHERE last LIKE 'w%' LIMIT 10;
..
+--------+----------+
10 rows in set (0.00 sec)


SELECT SQL_CALC_FOUND_ROWS id,username FROM users
WHERE last LIKE 'w%' LIMIT 10;
SELECT FOUND_ROWS();
...
10 rows in set (8.81 sec)

mysql> SELECT FOUND_ROWS();
+--------------+
| FOUND_ROWS() |
+--------------+
|         1671 |
+--------------+
1 row in set (0.02 sec)
```


Conclusion

CONCLUSION

- Essential tool for SQL analysis
- Not the only information you need
- Other Related Presentations
  - Understanding MySQL Indexes
  - Improving Performance with Better Indexes

Slides at http://j.mp/EM-Explain

$Em = ps^n$

Ronald Bradford
http://effectiveMySQL.com