PERCONA

EXPLAIN Demystified

Baron Schwartz
February 2012

Agenda

- What is EXPLAIN?
- How MySQL executes queries
- How the execution plan becomes EXPLAIN
- How to reverse-engineer EXPLAIN
- Advanced features and helpful tools
- Resources

What is EXPLAIN?

- Shows the estimated query execution plan
- Only for SELECT in MySQL 5.5 and previous

What is EXPLAIN?

```
mysql> explain select title from sakila.film where film_id=5\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: film
         type: const
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 2
          ref: const
         rows: 1
        Extra:
```

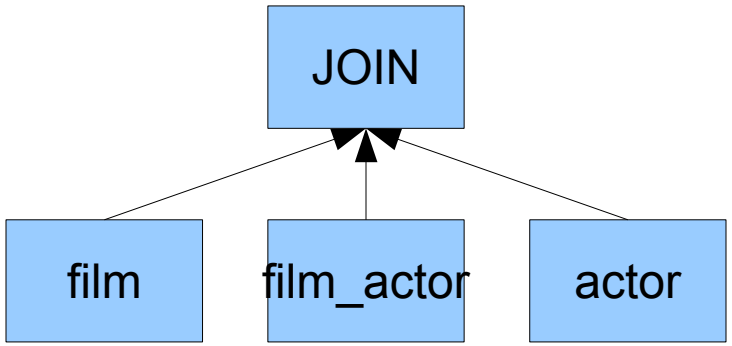
But First...

- How does MySQL execute queries?
- SQL => Parse Tree => Execution Plan
- Executor works from the plan
  - The plan is a data structure, not byte-code
  - The executor makes storage engine calls

The Execution Plan

SELECT... sakila.film
JOIN sakila.film_actor USING(film_id)
JOIN sakila.actor USING(actor_id)

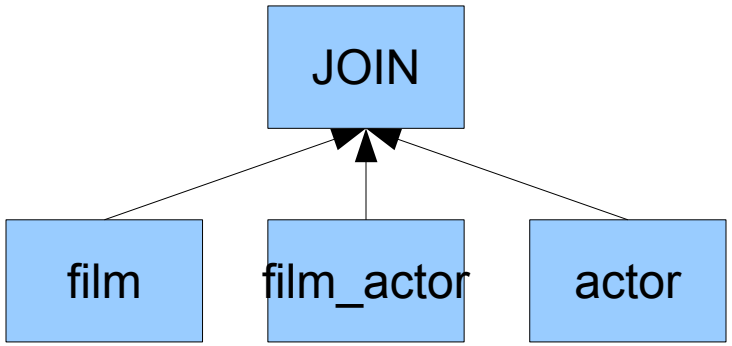
A possible strategy



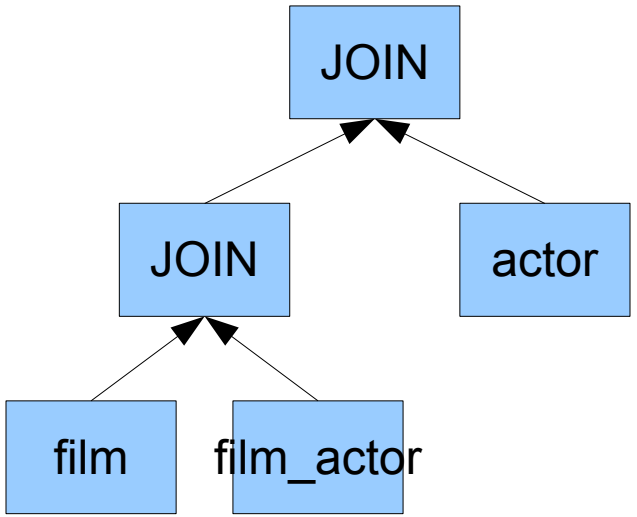
The Execution Plan

SELECT... sakila.film
JOIN sakila.film_actor USING(film_id)
JOIN sakila.actor USING(actor_id)

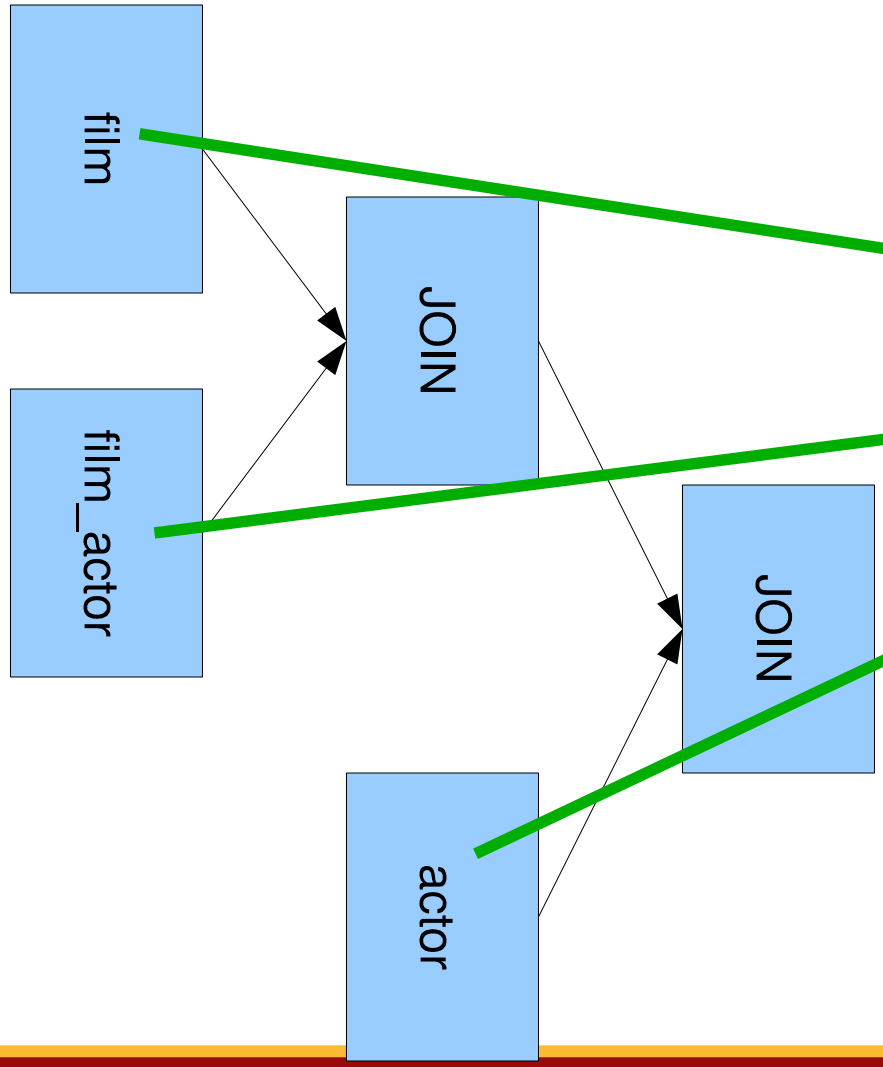
A possible strategy



MySQL's actual strategy



Where EXPLAIN comes from




```
mysql> EXPLAIN SELECT * FROM sakila.film
    -> JOIN sakila.film_actor USING(film_id)
    -> JOIN sakila.actor USING(actor_id);
+----+-------------+------------+--------+---------
| id | select_type | table      | type   | possible
+----+-------------+------------+--------+---------
|  1 | SIMPLE      | film       | ALL    | PRIMARY
|  1 | SIMPLE      | film_actor | ref    | PRIMARY,
|  1 | SIMPLE      | actor      | eq_ref | PRIMARY
+----+-------------+------------+--------+---------
```

Generating EXPLAIN

- MySQL actually executes the query
- But at each JOIN, instead of executing, it fills the EXPLAIN result set
- What is a JOIN?
  - Everything is a JOIN, because MySQL always uses nested-loops
  - Even a single-table SELECT or a UNION or a subquery

```
mysql> explain select title from sakila.film where film_id=5\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: film
         type: const
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 2
          ref: const
         rows: 1
        Extra:
```

The “id” Column

- id: which SELECT the row belongs to
  - If only SELECT with no subquery or UNION, then 1
  - Otherwise, generally numbered sequentially
  - Simple/complex types
    - simple: there is only one SELECT in the whole query
    - 3 subtypes of complex
      - subquery: numbered according to position in SQL text
      - derived: executed as a temp table
      - union: fill a temp table, then read out with a NULL id in a row that says UNION RESULT

The “id” in a Subquery

```
mysql> EXPLAIN SELECT
       (SELECT 1 FROM sakila.actor
        LIMIT 1)
       FROM sakila.film;
+----+--------------+-------+...
| id | select_type | table |...
+----+--------------+-------+...
|  1 | PRIMARY      | film  |...
|  2 | SUBQUERY     | actor |...
+----+--------------+-------+...
```

The “id” in a Derived Table

```
mysql> EXPLAIN SELECT film_id
       FROM (
       SELECT film_id FROM sakila.film
       ) AS der;
+----+-------------+-------------+...
| id | select_type | table       |...
+----+-------------+-------------+...
|  1 | PRIMARY     | <derived2>  |...
|  2 | DERIVED     | film        |...
+----+-------------+-------------+...
```

The “id” in a UNION

```
mysql> EXPLAIN SELECT 1
       UNION ALL SELECT 1;
+------+--------------+-------------+...
| id   | select_type  | table       |...
+------+--------------+-------------+...
|  1   | PRIMARY      | NULL        |...
|  2   | UNION        | NULL        |...
| NULL | UNION RESULT | <union1,2>  |...
+------+--------------+-------------+...
```

```
mysql> explain select title from sakila.film where film_id=5\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: film
         type: const
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 2
          ref: const
         rows: 1
        Extra:
```

The “select_type” Column

- Simple vs. complex; which type of complex
- Special UNION rule
  - First contained SELECT matches outer context
- Dependencies, uncacheability
  - Refers to the item_cache, not query cache

```
mysql> explain select title from sakila.film where film_id=5\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: film
         type: const
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 2
          ref: const
         rows: 1
        Extra:
```

The “table” Column

- The table name, or its alias
- Derived tables are complicated
  - Forward reference: <derivedN>
- UNION is complicated too
  - Backward references: <union1,2,3...>

Complicated Example

id	select_type	table
1	PRIMARY	<derived3>
3	DERIVED	actor
2	DEPENDENT SUBQUERY	film_actor
4	UNION	<derived6>
6	DERIVED	film
7	SUBQUERY	store
5	UNCACHEABLE SUBQUERY	rental
NULL	UNION RESULT	<union1,4>

Complicated Example

id	select_type	table
1	PRIMARY	<derived3>
3	DERIVED	actor
2	DEPENDENT SUBQUERY	film_actor
4	UNION	<derived6>
6	DERIVED	film
7	SUBQUERY	store
5	UNCACHEABLE SUBQUERY	rental
NULL	UNION RESULT	<union1,4>

Complicated Example

id	select_type	table
1	PRIMARY	<derived3>
3	DERIVED	actor
2	DEPENDENT SUBQUERY	film_actor
4	UNION	<derived6>
6	DERIVED	film
7	SUBQUERY	store
5	UNCACHEABLE SUBQUERY	rental
NULL	UNION RESULT	<union1,4>

The SQL, If You Want It

```
EXPLAIN
SELECT actor_id,
   (SELECT 1 FROM sakila.film_actor
      WHERE film_actor.actor_id = der_1.actor_id LIMIT 1)
FROM (
   SELECT actor_id
   FROM sakila.actor LIMIT 5
) AS der_1
UNION ALL
SELECT film_id,
   (SELECT @var1 FROM sakila.rental LIMIT 1)
FROM (
   SELECT film_id,
      (SELECT 1 FROM sakila.store LIMIT 1)
   FROM sakila.film LIMIT 5
) AS der_2;
```

The “type” Column

- How MySQL will access rows
- Worse to better:
  - ALL, index, range, ref, eq_ref, const, system, NULL

```
mysql> EXPLAIN SELECT ...
           id: 1
  select_type: SIMPLE
        table: film
         type: range
```

The Index-Related Columns

- possible_keys
  - Which indexes were considered?
- key
  - Which indexes did the optimizer choose?
- key_len
  - How many bytes of the index will be used?

```
mysql> EXPLAIN SELECT ...
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 2
```

The “ref” Column

- The source of values used for lookups

```
mysql> EXPLAIN
    -> SELECT STRAIGHT_JOIN f.film_id
    -> FROM sakila.film AS f
    ->    INNER JOIN sakila.film_actor AS fa
    ->       ON f.film_id=fa.film_id AND fa.actor_id = 1
    ->    INNER JOIN sakila.actor AS a USING(actor_id);
+-------+--------------------+---------+-----------------------+
| table | key                | key_len | ref                   |
+-------+--------------------+---------+-----------------------+
| a     | PRIMARY            | 2       | const                 |
| f     | idx_fk_language_id | 1       | NULL                  |
| fa    | PRIMARY            | 4       | const,sakila.f.film_id |
+-------+--------------------+---------+-----------------------+
```

The “rows” Column

- Estimated #rows to examine in the table/index

```
mysql> EXPLAIN SELECT * FROM sakila.film
    -> WHERE film_id > 50;
          rows: 511
         Extra: Using where
```

The “filtered” Column

- The percentage of rows that satisfy WHERE
- Usually 0 or 100; complex behavior

```
mysql> explain select title from sakila.film where film_id=5\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: film
         type: const
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 2
          ref: const
         rows: 1
        Extra:
```

The “Extra” Column

- Using index
- Using where
- Using temporary
- Using filesort
- Sort-merge types
- Lots more values; see the documentation

Using pt-visual-explain

```
JOIN
+- Unique index lookup
|  key             f->PRIMARY
|  possible_keys   PRIMARY
|  key_len         2
|  ref             sakila.film_actor.film_id
|  rows            1
+- JOIN
   +- Index lookup
   |  key             film_actor->PRIMARY
   |  possible_keys   PRIMARY,idx_fk_film_id
   |  key_len         2
   |  ref             sakila.actor.actor_id
   |  rows            13
   +- Index scan
      key             actor->PRIMARY
      possible_keys   PRIMARY
      key_len         2
      rows            200
```

EXPLAIN EXTENDED

- Lets you see how MySQL rewrites queries
- Use EXPLAIN EXTENDED <QUERY>
  - Then SHOW WARNINGS
- The output is too noisy to show here

EXPLAIN PARTITIONS

- Lets you see how MySQL prunes partitions

```
mysql> explain partitions select * from t\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: t
   partitions: p0,p1,pinf
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 3546
        Extra:
```

EXPLAIN PARTITIONS

- Lets you see how MySQL prunes partitions

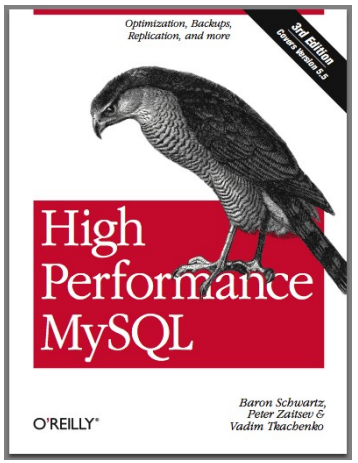
```
mysql> explain partitions select * from t where a between 10 and
99\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: t
   partitions: p1
```

Improvements in MySQL 5.6

- Doesn't materialize “derived tables” early
- EXPLAIN for non-SELECT
- Possibly an EXPLAIN that's not a “table”
- Optimizer trace
- More on MySQL 5.6:
  - http://goo.gl/YhybM
  - http://goo.gl/yJOn3

Resources

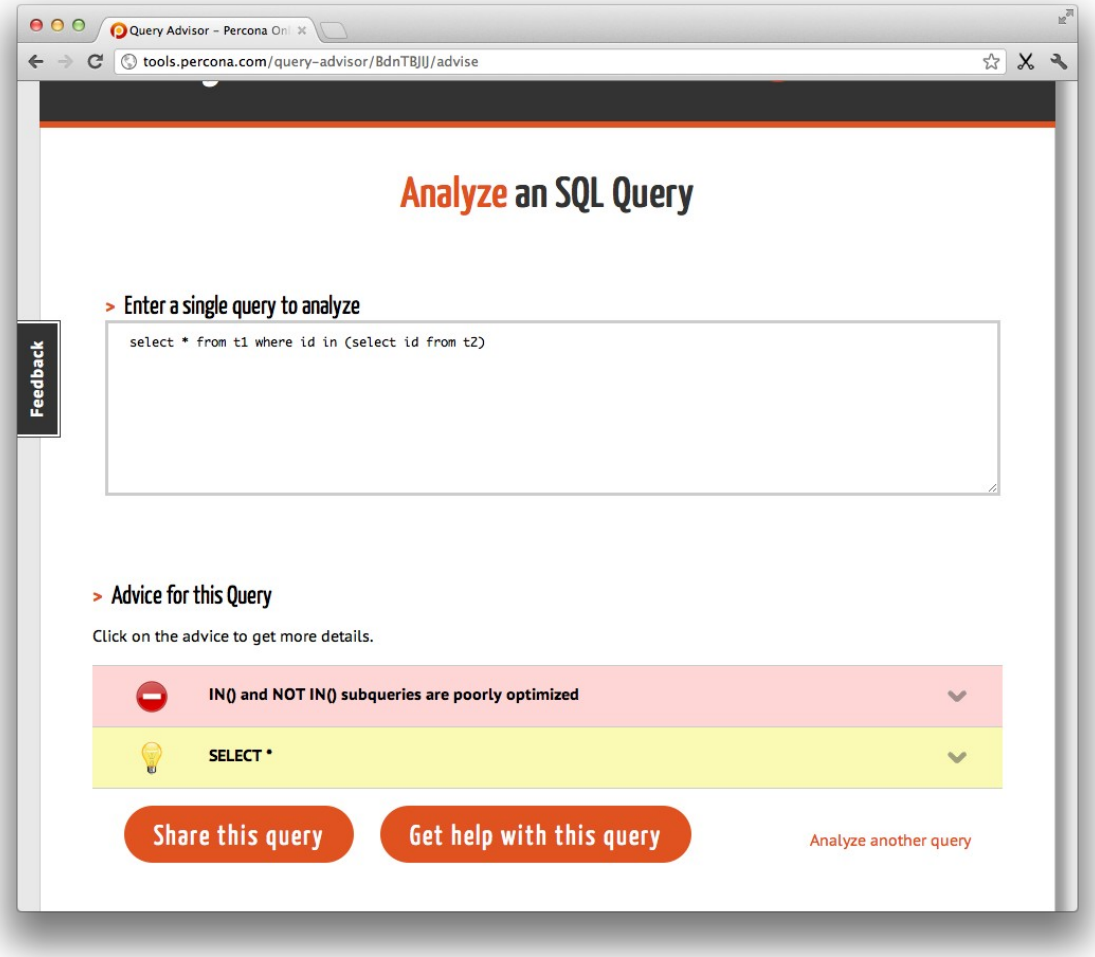
- High Performance MySQL
  - Third edition in a matter of weeks!
  - http://www.highperfmysql.com/
- Percona Training
  - http://www.percona.com/training/
- Percona Webinars
  - http://www.percona.com/webinars/
  - Mar 14: Optimizing Configuration
  - Apr 4: Percona XtraDB Cluster



Resources

- Percona Toolkit
  - pt-visual-explain tool makes EXPLAIN readable
  - http://www.percona.com/software/
- Percona Online Tools
  - http://tools.percona.com/
  - Generate my.cnf files easily
  - Get advice on SQL

Query Advisor Tool



MySQL Conference & Expo
April 10-12, Santa Clara

www.percona.com/live

PERCONA LIVE





baron@percona.com
@xaprb