Are You Getting the Best Out of Your MySQL Indexes?

Sheeri Cabral
Senior DB Admin/Architect, Mozilla
@sheeri  www.sheeri.com

Slides
What is an index?
KEY vs. INDEX

KEY = KEY CONSTRAINT

PRIMARY, UNIQUE, FOREIGN

Everything else is an “implementation detail”

The plural of INDEX is....?
More Lingo

Simple

(last_name)

Composite

(last_name,first_name)
B-TREE for InnoDB, MyISAM

Can be HASH for MEMORY
B-tree

(Image from Wikipedia)
B-trees Are Excellent For...

A range search - foo BETWEEN 5 AND 10
One equality match - foo=11
A few equality matches - foo IN (5,10,11)

- How is it done?
Index on \( (\text{last\_name}, \text{first\_name}) \)

Used find words beginning with “g” (last\_name)

Not used to find words ending with “g” (first\_name)

OK to have \( (\text{last\_name}, \text{first\_name}) \) and (first\_name)

Like a dictionary index

(Image from Wikipedia)
MySQL Uses Indexes For...

- Matching a WHERE clause
- Eliminating rows
- Resolving MIN() or MAX() values
- Eliminating sorting
- Helping with grouping
- Everything – Covering index
MySQL Ignores Indexes For...

Functions

\[ \text{DATE(ts\_col)=}'2012\_08\_11' \]

JOINs if the fields are not similar

\[ \text{date\_col}=\text{'2012\_08\_11 00:00:00'} \]
MySQL Ignores Indexes For...

Queries with multiple WHERE clauses
not all using the same index
joined by OR

For example, imagine a test table, with an index
on (last_name, first_name)
test,(last_name,first_name)

Queries that use the index:

SELECT * FROM test WHERE last_name='Cabral';

SELECT * FROM test
    WHERE last_name='Cabral' AND first_name='Sheeri';

SELECT * FROM test
    WHERE last_name='Cabral'
    AND (first_name='Tony' OR first_name='Antonio');
Queries that DO NOT use the index:

SELECT * FROM test WHERE first_name='Sheeri';

SELECT * FROM test
WHERE last_name='Cabral' OR first_name='Sheeri';
Composite Indexes

Index on (last_name,first_name,middle_name)

Functions as:
  (last_name,first_name,middle_name)
  (last_name,first_name)
  (last_name)
MySQL Ignores Indexes For...

“Too much” data, just do a full table scan (7,9,12)
6 lookups vs. walking the 10-node tree
“Too Much” Data

“Too much” is about 15-25%

How is that calculated?
Exact in MyISAM (writes are table-locking)
Approximate in InnoDB
“value group”
Average value group size
Used for approx rows for rows read, joins
Average Value Group Size

Body parts: 2 eyes, 10 fingers
Average value group size = 6
Not perfect optimization for either eyes or fingers
Estimate is longer than reality for eyes
Estimate is shorter than reality for fingers
Remember the “too much data” feature/problem
Composite Index

Like a sorted array
Can be ASC or DESC
But not ASC, DESC or DESC, ASC

(Image from Wikipedia)
NULL and Equality

NULL = x is not true for ANY value of x

NULL = NULL is not true

If a referenced value in an equality is NULL

MySQL immediately returns false
NULL and Equality

NULL-safe operator is $\leftrightarrow$

No NULL-safe inequality operator

$!(\text{foo} \leftrightarrow \text{bar})$

Remember the “too much data” feature/problem
NULL and Value Groups

Options - `innodb_stats_method`, `myisam_stats_method`

- `nulls_equal` (default)
- `nulls_unequal`
- `nulls_ignored`
PRIMARY Keys

Row identifiers

Cannot be NULL

InnoDB orders on disk in PRIMARY KEY order
UNIQUE Keys

Row identifiers

Can be NULL

Both PRIMARY/UNIQUE can be composite index
FOREIGN Keys

Parent/child relationship

e.g. payment->customer_id

Cascading update/deletes
### Numeric vs. Human-readable

#### customer_id | status_id
---|---
121 | 1
122 | 2
125 | 1

#### customer_id | status
---|---
121 | free
122 | paid
125 | free

#### status_id | status
---|---
1 | free
2 | paid
3 | disabled
Prefix Indexing

For strings

Up to 767 bytes on InnoDB, 1000 on MyISAM

Beware of charset!
FULLTEXT Indexing

No prefix indexing

Only for CHAR, VARCHAR, TEXT

MyISAM only until MySQL 5.6
GROUP BY and Sorting

By default, GROUP BY also sorts

May cause 'filesort' in EXPLAIN

ORDER BY NULL

If you do not care about the return order
Knowing All This...

Use EXPLAIN

If you are not getting the index you expect

Check your expectations

Or file a bug: http://bugs.mysql.com
How Do I Use EXPLAIN?


index_merge

On surface, looks good

Use more than one index

Better in MySQL 5.6
index_merge before 5.6

Indicates you could make a better index

Index merge not used when it should have been
  – e.g. if range scan was possible

May have merged more indexes than necessary
Questions? Comments?

OurSQL Podcast
www.oursql.com

MySQL Administrator's Bible
- tinyurl.com/mysqlbible

planet.mysql.com
mysqlmarinate.com