MySQL Tricks

Tips and Tricks from the Phorum Team
Brian Moon, Lead Developer - Phorum
2008 MySQL Conference & Expo
http://www.phorum.org/
What is Phorum?

• Open Source message board system
• In use on MySQL.com and Innodb.com
• Developers obsessive about performance

Come by our booth in the DotOrg Pavilion! Just look for the sleepy dog. DOH!
Quick Tips
SQL_COUNT_FOUND_ROWS

Saves the total row count a query matched NOT including any LIMIT clause. YMMV

Usage:

SELECT SQL_COUNT_FOUND_ROWS
from really_big_table
LIMIT 0, 20

select found_rows()
Extra Layers

Don’t use PHP based abstraction layers (PEAR::DB, ADOdb, etc.)

Remember this is a talk about speed. Calling an extra function before the real MySQL function is called will always be slower.

PDO is a C extension and is not slow.
Never Use Subqueries

• MySQL Subqueries are still a work in progress
• The inner query is run once for every row in the outer query
• Never say never? There is progress in this area. Stay tuned to the MySQL docs.
Be Careful With Joins

• Only join on indexed columns

• Do not use expression in your joins
  
  inner join t2 on t1.field1+10=t2.field2

• Very often, two queries are faster even with the programming overhead to merge them
Buffered vs. Unbuffered
What Are Buffered Queries?

- Data returned from the database is held in memory for use by other MySQL functions.
- Required for some MySQL functions to work. e.g. `mysql_num_rows`, `mysql_seek`
- This is the default behavior and most commonly used.
What Are Unbuffered Queries?

- Data is only sent from the server when it is requested.
- Some MySQL functions will not work. e.g. `mysql_num_rows`, `mysql_seek`.
- The results must be fully read and the result freed before another query can run.
When To Use Unbuffered Queries

Anywhere you have code like this:

```php
while($row = mysql_fetch_assoc($res)){
    $my_array[] = $row;
}
```

Which for me, is all over the place.

They are especially useful for large datasets.
When NOT To Use Unbuffered Queries

- Using mysql_num_rows() and other functions that use the buffered results
- Read from multiple result sets at one time
- Complex operations are done while reading data into user land
How to use Unbuffered Queries

• Older MySQL Extension offers the mysql_unbuffered_query function

• For MySQL Improved, pass MYSQLI_USE_RESULT to the third parameter of mysqli_query.

• For PDO, use PDOStatement::fetchAll()
You Can Pick Your Keys
Forcing your keys

- The MySQL optimizer has rules
- Sometimes, you need to break those rules
- You have to know your data
- You have to test a lot
select
    message_id, subject
from
    phorum_messages
where
    forum_id in (12,14,16,17,18,
                 19,20,21,22,23,24,25,26,28,
                 57,58,59,61,62,63,64,65,66,67)
    and status=2
    and moved=0
ORDER BY
    message_id DESC
limit 50;

50 rows in set (1.20 sec)
id: 1
select_type: SIMPLE
table: phorum_messages
type: ref
possible_keys: status_forum,foru.....
key: status_forum
key_len: 1
ref: const
rows: 33985
Extra: Using where; Using filesort
select
message_id, subject
from
phorum_messages use key (primary)
where
  forum_id in (12,14,16,17,18,
  19,20,21,22,23,24,25,26,28,
  57,58,59,61,62,63,64,65,66,67)
and status=2
and moved=0
ORDER BY
  message_id DESC
limit 50;

50 rows in set (0.00 sec)
********** 1. row **********
    id: 1
    select_type: SIMPLE
    table: phorum_messages
    type: index
    possible_keys: NULL
    key: PRIMARY
    key_len: 4
    ref: NULL
    rows: 67970
    Extra: Using where
I Know My Data

• The majority of the rows in the table match the status in the query.

• Also, the majority of the rows will match the large IN clause.

• So, we can scan on a fast key and most of the rows will match the other things in the where clause.
Test Your Queries

• If EXPLAIN looks wrong, try using FORCE INDEX with another key.

• Actually run the query. Sometimes EXPLAIN is confusing.

• Test how the query will scale when it is being run 100 times per second.
Full-Text > LIKE
Full-Text Search

- **Pros**
  - It is internal to MySQL
  - It is better than complicated LIKE queries

- **Cons**
  - Tricky with other indexes
  - Not all hosts enable it
  - Only works with MyISAM
Working With Full-Text

- Understand that MySQL will use only the FullText index for a query FullText queries
- Mixing other things into the where clause or ordering by ANY fields slows it down
- Returning loads of data to the application makes things slow and uses memory
- Phorum uses temporary heap tables to get around these issues
CREATE TEMPORARY TABLE foo ( KEY (message_id) ) ENGINE=HEAP
SELECT message_id
FROM   phorum_search
WHERE  MATCH (search_text)
AGAINST ('+some +search' IN BOOLEAN MODE);
SELECT SQL_CALC_FOUND_ROWS *
FROM phorum_messages
INNER JOIN foo USING (message_id)
WHERE status=2
and forum_id in (12,14,16,17,18,19,20,21,22,23,24,25,26,28,57,58,59,61,62,63,64,65,66,67)
and datestamp > 1200390957
ORDER BY datestamp DESC
LIMIT 0, 30;
Adding other searches

- Search by author, subject, etc.
- Create more temporary tables
- Merge them together in yet another table
- Yes, this really is fast (unless the MySQL server is badly configured)
Alternatives

- Use old fashioned LIKE clauses (Phorum has this for poor users with bad hosts)
- Use an external system
  - Sphinx Search - They are in the DotOrg Pavilion as well.
  - Lucene - Java based full text engine. There is a Zend class for this as well.
The current PHP mysql extensions use libmysqlclient.

This code is not “part of PHP”

Therefore, it does not have to follow the PHP rules

Things have to be copied over and over
mysqlnd, A New Hope

• mysqlnd “uses many of the proven and stable PHP internal functions”
• That means it respects memory limits
• It also means it can move data from MySQL directly into PHP data structures
• Available starting with PHP 5.3
# ./php-with-lib bigselect.php
All OK!

# ./php-with-nd bigselect.php

Fatal error: Allowed memory size of 2097152 bytes exhausted (tried to allocate 800000 bytes) in /home/brian/bigselect.php on line 14

The first script used the memory. You just have no control over it!
MySQL Tricks
Tips and Tricks from the Phorum Team
Brian Moon, Lead Developer - Phorum
2008 MySQL Conference & Expo
http://www.phorum.org/