ODTUG Kaleidoscope 2010, Session 10
Tuesday, June 29 - 4:45 PM - 5:45 PM

Using MySQL Meta Data Effectively

Dossy Shiobara, Panoptic.com

This presentation discusses what MySQL meta data is available including the 'mysql' meta schema, the `INFORMATION_SCHEMA` (`I_S`) tables first introduced in MySQL 5.0 and extended in MySQL 5.1, storage engine specific `INFORMATION_SCHEMA` tables, as well as techniques for writing your own `INFORMATION_SCHEMA` plug-ins. MySQL also provides a number of `SHOW` commands that provide easily formatted presentation of MySQL meta data. Dossy Shiobara will also discuss some of the limitations and performance implications of the `INFORMATION_SCHEMA`.

Tags: meta data, `INFORMATION_SCHEMA`, `SHOW` commands, mysql schema, plugins

INTRODUCTION

Traditionally, MySQL users have used the classic `SHOW` statements to access metadata, in addition to the 'mysql' meta-database. However, in MySQL 5.0, the `INFORMATION_SCHEMA` was introduced, finally providing a uniform SQL interface to this metadata. In subsequent versions, the `INFORMATION_SCHEMA` has grown to provide even more access to metadata.

This paper discusses what information is available through the `INFORMATION_SCHEMA` (sometimes abbreviated as `I_S`), the benefits of having such a mechanism and what one should be aware of when using it.

IN THE BEGINNING

Originally, in MySQL leading up to 5.0, system metadata was only accessed through tables in the 'mysql' database or through `SHOW` statements. Those familiar with MySQL have likely encountered the 'user' table in the 'mysql' database before, which stores the relevant data for MySQL's access privilege system, or the `SHOW TABLES` statement that lists the tables in the currently active database.

While this initial implementation was adequate for MySQL's capabilities at the time, there are limitations to this approach. Introducing new types of metadata meant introducing new keywords into the grammar of the query parser to expose it. This is additional code to write and maintain in `sql_yacc.yy` when extending the functionality of MySQL.

BEHOLD, THE INFORMATION_SCHEMA

Other database software has addressed this through metadata tables or read-only views; Oracle has its data dictionary, Sybase and SQL Server has its system catalog. Rather than come up with its own convention, MySQL has instead chosen to base its implementation around the non-free ANSI/ISO SQL:2003 standard Part 11 "Schemata"[1] along with its own extensions to accommodate MySQL-specific metadata.

WHY DO I NEED IT?

Suppose you're not interested in extending the functionality of MySQL yourself, and happy to let other developers deal with the work involved in extending the parser's grammar with new keywords. You're in the large majority of MySQL users. So, why is the information schema important and how can it be useful to you?

The `INFORMATION_SCHEMA` is useful because it provides an interface to the system's metadata that can be accessed through standard SQL statements. While you can write SQL queries that use `SHOW` statements, you cannot perform simple operations like joins on the results. You cannot take the result of `SHOW` statements and use them as values in a `CREATE TABLE` or `INSERT` statement to store the values back into the database. These are all things that you can do when you have a standard SQL interface to your metadata, but cannot do using MySQL's `SHOW` statements.

A standardized information schema should make it easier for software tools to support MySQL. Previously, they needed to explicitly support MySQL's special `SHOW` statement interface, but now they only need to accommodate MySQL's extensions to its information schema implementation. This wider opportunity for developers and DBAs to have access to better tool support increases the value and usefulness of MySQL as a database.

If you are a developer of a MySQL plugin, you can publish your own metadata using the `system_vars` member of your plugin descriptor. Then, you can access this data through the `INFORMATION_SCHEMA GLOBAL_VARIABLES` table.

SO, WHAT'S IN IT?

In every release of MySQL 5.x, more and more metadata is being published through the `INFORMATION_SCHEMA`. Starting with MySQL 5.0, the privilege system metadata that is traditionally found in the 'mysql' database—the 'user', 'db', 'host', 'tables_priv', 'columns_priv' and 'procs_priv' tables—is now accessible through the information schema. Additionally, the structure of the database—its tables, columns, views, index statistics, stored routines, constraints and triggers—is also available. In MySQL 5.1, information about plugins, partitions, events, files, processes, system status and variables were added. For a comprehensive view of all the information that is available in the `INFORMATION_SCHEMA`, you should consult the MySQL documentation on it.[2]

Suppose we wanted to find all of the `TEXT` columns in the current database. We can now do this with a single query:

```
mysql> SELECT table_name, column_name, column_type
       FROM information_schema.columns
       WHERE table_schema = DATABASE()
       AND column_type LIKE '%text%';

+-------------+-------------------------+-------------+
| table_name  | column_name             | column_type |
+-------------+-------------------------+-------------+
| COLUMNS     | COLUMN_DEFAULT          | longtext    |
| COLUMNS     | COLUMN_TYPE             | longtext    |
| EVENTS      | EVENT_DEFINITION        | longtext    |
| PARTITIONS  | PARTITION_EXPRESSION    | longtext    |
| PARTITIONS  | SUBPARTITION_EXPRESSION | longtext    |
| PARTITIONS  | PARTITION_DESCRIPTION   | longtext    |
| PLUGINS     | PLUGIN_DESCRIPTION      | longtext    |
| PROCESSLIST | INFO                    | longtext    |
| ROUTINES    | ROUTINE_DEFINITION      | longtext    |
| TRIGGERS    | ACTION_CONDITION        | longtext    |
| TRIGGERS    | ACTION_STATEMENT        | longtext    |
| VIEWS       | VIEW_DEFINITION         | longtext    |
+-------------+-------------------------+-------------+
```

Previously, you would have to iterate over each table in the database and use `SHOW COLUMNS`, one at a time.

Want a quick way to get a list of table names that could use an `OPTIMIZE TABLE` performed on them? Try this:

```
mysql> SELECT table_name
       FROM information_schema.tables
       WHERE table_schema = DATABASE()
       AND data_free > 0;
```

How about all the SQL stored procedures that contain the phrase "FROM tablename" in its procedure body?

```
mysql> SELECT routine_name
       FROM information_schema.routines
       WHERE routine_definition
             LIKE '%FROM tablename%';
```

These are just a few simple examples of what we can now do using MySQL's information schema.

THIS SOUNDS GREAT. WHAT'S THE CATCH?

MySQL's implementation of its information schema, which you can find in `sql/sql_show.cc`, can result in fairly expensive operations. Retrieving certain data from the information schema is done completely in-memory, but others, such as enumerating table columns, can result in MySQL needing to perform disk I/O in order to build the metadata in response. This can result in a very expensive query.

What's worse, some queries will acquire a process-wide lock (`LOCK_open`) around sections of the code that fulfills certain information schema requests, at least in MySQL 5.1. It is because of this that I recommend users be cautious about the kind and frequency of information schema queries they make, especially in a live, production environment.

BUT NOT ALL IS LOST!

However, don't let this discourage you! MySQL has a long history of gradual improvements over time, and given the usefulness of the information schema, optimizations will continue to be made. Learn how to navigate it and what valuable information you can get out of it now. When the performance issues are resolved, you will be prepared to take full advantage of them.

ACKNOWLEDGEMENTS

I'd like to thank everyone who has contributed to make MySQL what it is today; the ODTUG Kaleidoscope 2010 conference organizers, particularly Ronald Bradford, for giving me the opportunity to write this paper and present it; Sheeri K. Cabral for her continued encouragement and support.

I also want to thank my wife, Samantha, and daughters Charlene and Suzanne. Without them, none of this would matter.

REFERENCES

[1] MySQL 5.0 - Chapter 19. INFORMATION_SCHEMA Tables, http://dev.mysql.com/doc/refman/5.0/en/information-schema.html

[2] MySQL 5.5 - Chapter 19. INFORMATION_SCHEMA Tables, http://dev.mysql.com/doc/refman/5.5/en/information-schema.html