Thursday, July 25, 13

- Assumptions about attendees
  1) Working Developers
     - know how to write queries
     - probably have installed
     - worked with MySQL on a basic level (DML/DDL)
  2) Limited knowledge of MySQL
     - If you are more advanced then this, please feel you can leave to go to another talk.

- I will not be going into great detail on things since many of the areas/subjects or even commands I touch on could be a talk all by themselves. This is more to expose you to what is available and for you to later explore them in more detail as you have time or need them.
- I will be talking about the things that *I* feel are the most important for a new admin to know. Others may and probably will disagree. Please email me and let me know what *you* think. What did I forget? What did I talk about that should have been skipped or not spent so much time on? Any other feedback is very welcome
- I will also be including the latest features/options which I know not everyone is using. I try to point out versions if it is after 5.1
Agenda

• Access Control
• Metadata
• Server State
• Logs
• Backups
Access Control
Access Control

- 2 stage
  - Stage 1 - connecting
    - Who are you?

- server accepts or rejects the connection based on your identity

Who are you is answered by looking at the user account you are trying to connect with.
User Accounts

• 2 parts
  • username
  • host
  • Ex:
    • ‘root’@‘localhost’
    • ‘@‘127.0.0.1’

• CREATE USER
  • IDENTIFIED BY

```
mysql> CREATE USER 'lig'@'localhost' IDENTIFIED BY 's3cr3T';
Query OK, 0 rows affected (0.19 sec)
```

**Bullet 1:**
*username is either a blank value (anonymous account) or a non-blank value that is matched exactly.
*host name part can take many forms and can use wildcard characters*

**Bullet 2:**
Other than the initial accounts that are created when you install MySQL
*CREATE USER only creates the account – it does not provide any privileges
*IDENTIFIED BY
  - left off – no password
  - plain text – assigns password
  - can also be used to specify a specific authentication plugin (more on that later)*

username is for MySQL only – not Unix/windows/Linux
Ok - now that we understand more about user accounts,

Stage 1: matched against Host.
- This is checked first
- user table is read into memory and *SORTED* based on Host.
  Most specific Host values first (hostname/IP address) -> least specific last (%).
- the Host value is then matched
Access Control
continued

• 2 stage
  • Stage 1 - connecting
    • Who are you?
      • host
      • user

Then the User name is sorted (specific names first –> anonymous name last.

  – first match made (matching first the host, and then the user) is the user account that is used.
  – Gotcha – anonymous user on localhost will be chosen before a specific user with a wildcard host.
Access Control continued

• 2 stage
  • Stage 1 - connecting
    • Who are you?
      • host
      • user
    • Prove it!

– once a user/host combination is matched – check the password. (Blank password \(\neq\) Any password)
Passwords

- **SET PASSWORD**
- **Methods**
  - Original
  - 4.1
  - **SHA 256 (5.6)**

```sql
mysql> CREATE USER 'sha256user'@'localhost' IDENTIFIED WITH sha256_password;
Query OK, 0 rows affected (0.00 sec)

mysql> SET old_passwords = 2;
Query OK, 0 rows affected (0.00 sec)

mysql> SET PASSWORD FOR 'sha256user'@'localhost' = PASSWORD('sha256P@ss');
Query OK, 0 rows affected (0.09 sec)
```

**Methods:**

Original: 16-byte string

4.1: lots of changes made for better security including:

- making the column 41 bytes long
- change in the default hashing method

SHA 256: new to 5.6.6 – Optional

- uses the built-in sha256_password plugin

Note: 5.6 has new features/plugins for controlling password expiration and password strength. If you are interested be sure to look into them.
Access Control
continued

• 2 stage
  • Stage 1 - connecting
    • Who are you?
      • host
      • user
    • Prove it!
  • Stage 2 - request
    • For each request
      • What are you doing?
      • Are you allowed to do that?

Stage 2: Lots of the mysql db tables are used
  – This is where the GRANTS for a user come into affect.
GRANT

• Defines privileges and account characteristics
• Multiple privileges
  • Ex: SUPER, CREATE, ALTER, SELECT
• Multiple levels
  • Ex: Global, Database, Table, Column
• Account characteristics
  • WITH / REQUIRE

```
mysql> GRANT SELECT, INSERT, UPDATE ON mydb.tbl1 TO 'someuser'@'somehost' WITH GRANT OPTION REQUIRE SSL;
Query OK, 0 rows affected (0.24 sec)

mysql> GRANT SELECT (col1), INSERT (col1,col2) ON mydb.tbl2 TO 'someuser'@'somehost';
Query OK, 0 rows affected (0.24 sec)
```
REVOKE

- Removes privileges GRANTed
  - Does not extrapolate
- Doesn’t remove the user
- no host given
  - % is used
  - Does not extrapolate

```sql
mysql> SHOW GRANTS FOR ''@'localhost';
+----------------------------------------------------+
| Grants for @localhost                              |
| GRANT USAGE ON *.* TO ''@'localhost'               |
| GRANT ALL PRIVILEGES ON `test`.* TO ''@'localhost' |
+----------------------------------------------------+
2 rows in set (0.00 sec)
```

```sql
mysql> REVOKE DELETE ON test.t1 FROM ''@'localhost';
ERROR 1147 (42000): There is no such grant defined for user '' on host 'localhost' on table 't1'
```

Thursday, July 25, 13

1) Only *EXACTLY* the privileges GRANTed can be revoked.
   - Ex:
     ```sql
     mysql> SHOW GRANTS FOR ''@'localhost';
     +----------------------------------------------------+
     | Grants for @localhost                              |
     | GRANT USAGE ON *.* TO ''@'localhost'               |
     | GRANT ALL PRIVILEGES ON `test`.* TO ''@'localhost' |
     +----------------------------------------------------+
     2 rows in set (0.00 sec)
     ```
     ```sql
     mysql> REVOKE DELETE ON test.OpKreditoren FROM ''@'localhost';
     ERROR 1147 (42000): There is no such grant defined for user '' on host 'localhost' on table 'opkreditoren'
     ```

2) You need to use DROP USER to do that

3) ```sql
   mysql> select host from mysql.user where user='';
   +------------------------+
   | host                   |
   +------------------------+
   | Ligs-MacBook-Pro.local |
   | localhost              |
   +------------------------+
   2 rows in set (0.01 sec)```
• Superuser account
• Initially
  • Password empty
  • Localhost & sometimes wildcard
• Recommendations
  • Consider getting rid of the name ‘root’
  • Strong password \(\text{SET PASSWORD}\)
  • Limit to only localhost
  • Never used by an app

Recommendations:
1) give it some other name then root. Bob/Jane/Mary_Poppins etc (CREATE new user with full privileges, DROP root account)
2) give it a strong password – upper, lower, numeric, special chars, long
   – \text{SET PASSWORD}
   – shell> mysqladmin –u root password ‘newpass’
   – UPDATE mysql.user SET password
3) Limit it so root can only come from someplace you control.
4) Applications should have their at least 1 if not more users to access the DB depending upon what they want to do. Follow the principle of least privileges. Ex: A user that only reads, a user that can only INSERT.
So what is Metadata? Basically it is “Data about Data”.

What kind of metadata is available in MySQL? Ex: name of a database or table, storage engine of a table, the data type and or character set/collation of a column, or access privileges.

AKA: data dictionary or system catalog

MySQL provides access to metadata in 2 ways.
SHOW

- MySQL Specific
- “Old” way
- Information on
  - Databases
  - Tables
  - Columns
  - or Status info
- Metric crap-ton of them

Thursday, July 25, 13

For those of us that have been doing this for a bit – this is what we “grew up” on. I personally still tend to go first to this method.

- DB: SHOW DATABASES, SHOW CREATE DATABASES, SHOW PLUGINS, SHOW TRIGGERS, SHOW ERRORS, SHOW WARNINGS
- Tables: SHOW TABLES, SHOW CREATE TABLES, SHOW INDEX, SHOW CREATE TRIGGER, SHOW CREATE PROCEDURE
- Columns: SHOW COLUMNS,
- Status: SHOW MASTER STATUS, SHOW SLAVE STATUS, SHOW FULL PROCESSLIST, SHOW ENGINE INNODB STATUS

There are 41 different SHOW commands.
| SHOW DATABASES; | SHOW TRIGGERS; | SHOW ERRORS; |
| SHOW PLUGINS; | SHOW WARNINGS; | SHOW TABLES; |
| SHOW INDEX; | SHOW COLUMNS; | SHOW CREATE PROCEDURE; |
| SHOW CREATE TABLES; | SHOW CREATE DATABASES; | SHOW CREATE TRIGGER; |
| SHOW MASTER STATUS; | SHOW SLAVE STATUS; | SHOW FUNCTION STATUS; |
| SHOW ENGINE INNODB STATUS; | SHOW PROCEDURE STATUS; | SHOW FULL PROCESSLIST; |

Thursday, July 25, 13

Many have a ‘LIKE <pattern>’ part as well as a ‘WHERE’ clause to provide additional flexibility

SHOW PROCEDURE STATUS WHERE DB='test'
SHOW FUNCTION STATUS WHERE DB='test'
SHOW TABLES LIKE <pattern>
• “Virtual” Database
• Familiar to how “others” do it
• Use standard SELECT to query
• Can only see what you have access to
• Performance considerations

Thursday, July 25, 13
- **Introduced in 5.0.** Each version since then has added/improved what is there. Depending on what version you have, your mileage may vary.
- Technically it is made of read only views
- You can use standard SQL to query the I_S tables. Don’t have to remember specific SHOW commands for things.
- Each MySQL user has the right to access these tables, but can see only the rows in the tables that correspond to objects for which the user has the proper access privileges.
- touching more than 1 DB – check time and performance of the query.
<table>
<thead>
<tr>
<th>CHARACTER_SETS</th>
<th>COLUMNS</th>
<th>ENGINES</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVENTS</td>
<td>GLOBAL_STATUS</td>
<td>PARTITIONS</td>
</tr>
<tr>
<td>PROCESSLIST</td>
<td>ROUTINES</td>
<td>TABLES</td>
</tr>
<tr>
<td>TRIGGERS</td>
<td>VIEWS</td>
<td>STATISTICS</td>
</tr>
<tr>
<td>OPTIMIZER_TRACE</td>
<td>PLUGINS</td>
<td>INNODB_TRX</td>
</tr>
<tr>
<td>INNODB_LOCKS</td>
<td>INNODB_LOCK_WAITS</td>
<td>COLLATIONS</td>
</tr>
</tbody>
</table>

Thursday, July 25, 13
Listing taken from 5.6
Server State
Current Server Settings

Where to look

- Option file - my.cnf/my.ini
- Global Variables
  - SHOW GLOBAL VARIABLES;
  - SELECT * FROM INFORMATION_SCHEMA.GLOBAL_VARIABLES;

This lets us look at the current settings of the server. So if you need to find out how big your innodb buffer pool is or if the query cache is on or just what version you are running – this can all be found here.

And yes – you do need both. Some things in the my.cnf file are not reflected in the global variables. EX: --read-only, --replicate-* and --binlog-*

I_S.Global_variables introduced in 5.1
<table>
<thead>
<tr>
<th>Variable_name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto_increment_increment</td>
<td>1</td>
</tr>
<tr>
<td>auto_increment_offset</td>
<td>1</td>
</tr>
<tr>
<td>autocommit</td>
<td>ON</td>
</tr>
<tr>
<td>automatic_sp_privileges</td>
<td>ON</td>
</tr>
<tr>
<td>back_log</td>
<td>80</td>
</tr>
<tr>
<td>basedir</td>
<td>/Users/ligaya/</td>
</tr>
</tbody>
</table>

```
mysql> SHOW GLOBAL VARIABLES;
+---------------------------------+---------------------+
| Variable_name                    | Value               |
+---------------------------------+---------------------+
| auto_increment_increment         | 1                   |
| auto_increment_offset            | 1                   |
| autocommit                       | ON                  |
| automatic_sp_privileges          | ON                  |
| back_log                         | 80                  |
| basedir                          | /Users/ligaya/      |
```

**mysql_installs/mysql-5.6.12-osx10.7-x86_64**

- big_tables: OFF
- bind_address: *
- binlog_cache_size: 32768
- binlog_checksum: CRC32
- binlog_direct_non_transactional_updates: OFF
- binlog_format: STATEMENT
- binlog_max_flush_queue_time: 0
- binlog_order_commits: ON
- binlog_row_image: FULL
- binlog_rows_query_log_events: OFF
- binlog_stmt_cache_size: 32768
- bulk_insert_buffer_size: 8388608
- character_set_client: latin1
- character_set_connection: latin1
- character_set_database: latin1
- character_set_results: binary
- character_set_filesystem: binary

Thursday, July 25, 13

Taken from one of my test systems
What has the server done?

- **SHOW GLOBAL STATUS**
- **SELECT * FROM INFORMATION_SCHEMA.GLOBAL_STATUS**
  - Lists server counters
  - What the server is actually doing - not what you think it is
    - Uptime, Uptime_since_flush_status
    - Com_select, Com_insert, Com_update, Com_delete
    - Slow_queries
    - Created_tmp_tables, Created_tmp_disk_tables
    - Select_full_join
    - Table_locks_waited

---

Thursday, July 25, 13

- Slow_queries: # that have taken more than long_query_time seconds regardless if the slow query log is on.
- Select_full_join: # of joins that perform table scans because they do not use indexes.

I_S.Global_variables introduced in 5.1
```sql
mysql> SHOW GLOBAL STATUS;
+---------------------------------------+--------------+
| Variable_name                        | Value        |
+---------------------------------------+--------------+
| Aborted_clients                      | 569          |
| Aborted_connects                     | 185          |
| Binlog_cache_disk_use                | 0            |
| Binlog_cache_use                     | 0            |
| Bytes_received                       | 19565585748  |
| Bytes_sent                           | 684330624413 |
| Com_admin_commands                   | 450019       |
| Com_assign_to_keycache               | 0            |
| Com_alter_db                         | 0            |
| Com_alter_db_upgrade                 | 0            |
| Com_alter_event                      | 0            |
| Com_alter_function                   | 0            |
| Com_alter_procedure                  | 0            |
| Com_alter_server                     | 0            |
| Com_alter_table                      | 73319        |
| Com_alter_tablespace                 | 0            |
| Com_analyze                          | 6            |
| Com_backup_table                     | 0            |
| Com_begin                            | 27737        |
| Com_binlog                           | 0            |
| Com_call_procedure                   | 0            |
| Com_change_db                        | 4739958      |
| Com_change_master                    | 0            |
```

Taken from a server that is not one of test systems.

Thursday, July 25, 13
What is going right now?

- **SHOW FULL PROCESSLIST**
- **SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST\G**
- Shows connection information

```
mysql> show full processlist\G
*************************** 1. row ***************************
  Id: 3
  User: root
  Host: localhost
  db: mysql
  Command: Query
  Time: 0
  State: init
  Info: show full processlist

*************************** 2. row ***************************
  Id: 4
  User: ligaya
  Host: localhost
  db: test
  Command: Sleep
  Time: 88
  State:                  
  Info: NULL
2 rows in set (0.00 sec)
```

I_S.PROCESSLIST introduced in 5.1
Monitoring

• Do it!
• Why?
  • Baseline
  • Records changes
  • Identify trends
• Some options
  • MySQL Enterprise Monitor (MEM)
  • Cacti
  • Nagios

Thursday, July 25, 13

I don’t care what you use to monitor the MySQL server – you just have to have something doing it.

Why:
– You can’t notice changes in your server if you do not know what is “normal”.
– To diagnose what is wrong you need to know what changed and how it changed. Admittedly you may not have all the data needed from the monitoring, but something is better then nothing and it might help you decide what additional information to collect “next time”.
– Does an increase in connections signal a potential performance degradation? Are there temporal cycles – Per day/per year? Are you suddenly getting more writes to the system?

Options:
– MEM – D’uh – I work for MySQL. Available if you are a customer or if you just want to try it for free you can download it from Oracle’s “Software Delivery Cloud” AKA edelivery. The trial has no kill switch on it.
– Cacti and Nagios – I know Percona has Monitoring Plugins for these and I think they are open source – but I can’t guarantee that. I highly recommend you take a look at them and maybe talk to the great people at Percona about what is required to use them if you prefer to go this way.

*Note: Data Dog – one of the exhibitors
Logs
I recommend always explicitly setting the error log. Never wipe out the error log, many problems just can not be diagnosed without it. Instead archive old ones as needed. The error log history can give context to a current problem.

If you have any problems – or if you don’t – check out the error log.
2) You can get your greatest performance boosts from tuning your queries. Faster queries mean less contention for resources and locks... not to mention they are just faster.

3a) If you turn on the slow query log (slow-query-log=1 or no value), by default a file will be written named `host_name-slow.log` in the data directory.

4a) You can change the path and name with `--slow-query-log-file`.

4b) `--log-output` takes TABLE, FILE or NONE in a comma separated list of 1-N values. NONE overrides all. If TABLE is used, it will be written to the slow_log table in the mysql database.

4c) minimum requirements to be met for a query to make it into the Slow Query Log.

4d) Self explanatory. Though if you have a lot – you can throttle it with `--log_throttle-queries-not-using-indexes` (5.6)

5) To make sense of the slow query log you will want to use something like mysqldumpslow or Percona’s pt-query-digest.
General Query Log

• General Record
• Why?
  • Order is important
  • Exact query that came in
  • audit what a connection did
• Enable
  • Dynamically or with --general-log
• Options
  • --general-log-file (optional)
  • --log-output

Thursday, July 25, 13

1) writes information when clients connect and disconnect and writes the SQL received as they come in – not as they are executed. Can quickly take up a lot of space so pay attention.
2a) deadlock issues? unexpected results?
2b) is the app generating a query different then you thought? an unexpected value?
2c) what connection tried to do what on the server?
3) by default a file will be written named host_name.log in the data directory.
4a) 5.1 Set the path and name of the general log file
4b) 5.1 --log-option takes TABLE, FILE or NONE in a comma separated list of 1–N values. NONE overrides all. If TABLE is used, it will be written to the general_log table in the mysql database.
Binary Log

• Logs database change events
• Why?
  • Replication
  • Data recovery
• Enable
  • --log-bin
• Options
  • Lots & Lots & Lots
• mysqlbinlog

Thursday, July 25, 13

1) logs changes to the database – ALTER, UPDATE, CREATE, INSERT, UPDATE, DELETE, etc. – Not SELECT or SHOW.
2a) Integral part of how replication works. Replication is again one of those subjects that are an entire talk all by itself. So I’m just going to leave it alone
2b) People forget however that the binary log can also be used for data recovery. Assuming you have a viable backup of the system, you can re-execute the events in the binary log from the time of the backup to the present or a specific point in time.
3) By default the value of the option ‘pid-fle’ (default is hostname) is concatenated with –bin to give a basename. A numeric extension is appended to the binary log basename to generate the binary log file names. The number is increased every time the server starts a new file. These files by default are written to the datadir. You can alter the default behavior by giving a name and path.
4) There is a very long manual page on all the various binary log options and variables. Again if you want to know more please see the manual.
5) Getting human readable information from the binary log is best done with mysqlbinlog. The settings used for the binary log will dictate your usage of mysqlbinlog
Backups
Backup Types
Logical

- Saved
  - Logical structure
  - Content
- Machine Independent
- Slower
- Server Up/Warm
- Full Granularity

1) backup does not include log and/or configuration files or other db related files that are not part of DBs
1a) CREATE database/tables/procedure/functions
1b) INSERT or delimited-text files
3) Pull data out of the server, and write it down in logical format. Load it back in.
5) Backup and restore granularity is available at the server level (all databases), database level (all tables in a particular database), or table level. This is true regardless of storage engine.
Physical

• Raw copies
• Faster
• Compact
• File based granularity
• Server down or Locked

Thursday, July 25, 13

1) exact copies of the database directories and files. Typically all or part of the data directory. Portable only to other machines that have identical or similar hardware characteristics. Physical backup tools include the `mysqlbackup` of MySQL Enterprise Backup for InnoDB or any other tables, file system-level commands (such as `cp`, `scp`, `tar`, `rsync`), or `mysqlhotcopy` for MyISAM tables.

2) only copying files – not extraction and writing.

4) Backup and restore granularity is available at the server level (all databases), database level (all tables in a particular database), or table level. This is true regardless of storage engine. backup can include any related files such as log or configuration files.

5) Backups can be performed while the MySQL server is not running. If the server is running, it is necessary to perform appropriate locking so that the server does not change database contents during the backup. MySQL Enterprise Backup does this locking automatically for tables that require it.
Backup Methods
mysqldump

- Logical backup
- Commonly used
- Editable text files
- Very flexible
- Doesn’t scale well

Thursday, July 25, 13
- Commonly used for small systems and what most people use when they first start taking backups.
- Once you start getting some size to the system it can take a while to take the backup and restore it which may not be acceptable.
Filesystem Snapshot

- Physical backup
- OS Dependent
  - FLUSH TABLES WITH READ LOCK
  - Take the snapshot
  - UNLOCK TABLES
  - copy files over
  - startup instance and recover

This is not possible to do for everyone but it is fairly commonly done for those with larger systems.
Other Alternatives

- **MySQL Enterprise Backup**
  - Available to Enterprise customers
- **Percona’s Xtrabackup**
  - Free, open source and available through them
  - Heard lots of good things
- **Others**
  - Zmanda Recovery Manager for MySQL
  - mysqlhotcopy
  - Delimited text files

---

Thursday, July 25, 13

- MEB – D’uh – I work for them. Seriously though, it is a good way to go. It does a “hot” physical backup that puts the InnoDB storage engine into a consistent state.
- Xtrabackup – from what I understand it does basically the same thing as MEB... though I am sure there are difference in some of the exact capabilities.
- Others:
  - Zmanda is another backup system that I have heard of – but not a lot about... least not as much as Xtrabackup. Not really sure how good it is or exactly how it works but wanted to make sure you knew of it.
  - mysqlhotcopy is a perl script that does a physical backup of MyISAM and ARCHIVE tables
  - Delimited text files is another way to do a logical backup. Basically you just do a SELECT * INTO OUTFILE... setting up the delimiters as you desire. You reload it with a LOAD DATA INFILE.
That was a whirlwind review of some of the areas that I think are the most important for a beginner MySQL Admin to learn about.

There was a lot of information in this talk that you hopefully will go back to later and review for a better understanding.
Thursday, July 25, 13

That was a whirlwind review of some of the areas that I think are the most important for a beginner MySQL Admin to learn about.

There was a lot of information in this talk that you hopefully will go back to later and review for a better understanding.