

MySQL : MySQL Competency

This page last changed on Feb 11, 2007 by [admin](#).

MySQL Intro

Why?

MySQL is free, open source based relational database server. It has support for ACID compliance, stored procedures, functions, triggers, and features many storage engines based on intended usage. MySQL is used by organizations such as Flickr, Ampd Mobile, Digg, and many other large internet sites.

Versions

3.23 - First major version to be shipped and widely used on the internet.

4.1 - Added transactional ACID support via InnoDB storage engine.

5.0 - Added stored procedure and other programming support as well as cluster engine NDBCLUSTER

5.1 - Beta, adds many new features to NDBCLUSTER engine and higher performance locking.

Files

Startup and Administrative Scripts

```
/etc/init.d/mysqld {start|stop|restart|status}
```

- Use this script to perform the functions of the server
- To have MySQL start at boot time (init 3) make sure ``chkconfig mysqld on`` has been run from the command line.

Data Files and Directory Structure

- RPM install:
data dir: `/var/lib/mysql`
along with various shared libs and binaries distributed in proper OS locations
- Binary install:
data dir: `/usr/local/mysql/data`
shared libs and binaries `/usr/local/mysql`
- Databases
Databases are kept in directories in the main data directory.
Every MySQL server will have a "mysql" database in the data directory where user permissions, host information, and many other vital parts are stored. For a corrupted or new install the `mysql_install_db` script will be run to generate this database.

Configuration

- Main file: /etc/my.cnf
 - User specific client configuration files are stored in \$HOME/
-

Database Engines

The typically used engines include the following.

InnoDB

<http://www.innodb.com/index.php>

InnoDB is the most popular transactional storage engine for MySQL. InnoDB provides full support of standard SQL isolation levels for ACID-compliant transaction processing, and maximizes throughput with row level locking and multi-version concurrency control (MVCC). InnoDB provides high performance by using many techniques to minimize expensive disk I/O through the efficient use of memory and processor resources. InnoDB ensures data integrity and reliability, with such capabilities as referential integrity support and automatic data recovery following hardware and software failures.

InnoDB is developed by Innobase Oy, a subsidiary of Oracle, and a premier partner of MySQL AB. MySQL AB distributes InnoDB in both MySQL Enterprise and the MySQL Community Edition. As part of MySQL Enterprise subscription, users can obtain support for InnoDB through MySQL AB. In April 2006, MySQL and Innobase Oy agreed to a multi-year extension of their licensing agreement.

InnoDB fully integrates with MySQL 5.0 and its new features such as stored procedures, triggers and views. In addition, InnoDB in MySQL 5.0 introduced a compact table format that saves about 20 % of space when compared to the old table format. In 5.0, InnoDB also supports two-phase commit through the XA protocol.

MyISAM

<http://en.wikipedia.org/wiki/MyISAM>

MyISAM is the default storage engine for the MySQL relational database management system. It is based on the older ISAM code but has many useful extensions. In recent MySQL versions, the InnoDB engine has widely started to replace MyISAM due to its support for transactions, referential integrity constraints, and higher concurrency.

Each MyISAM table is stored on disk in three files. The files have names that begin with the table name and have an extension to indicate the file type. A .frm file stores the table definition. The data file has a .MYD (MYData) extension. The index file has a .MYI (MYIndex) extension.

NDBCLUSTER

MySQL Cluster is a high-availability, high-redundancy version of MySQL adapted for the distributed computing environment. It uses the NDB Cluster storage engine to enable running several MySQL servers in a cluster.

It requires several servers to operate most efficiently. One management node, one or more data nodes, and one or more sql nodes. It is recommended to use heartbeat and ldirector to access the cluster via a VIP IP Address.

The cluster engine, as of 5.0.x versions has limitations such as no foreign key support, all database information must be able to fit into RAM, and locking contention is limited to READ-COMMITTED.

Memory

The Memory Engine is a highly useful engine for temporary tables/schemas that are generated by SP's (Stored Procedures) or other means. It is faster in many cases than other engines but the data will only be held in memory while the server is on. As soon as the server is off the data disappears.

Logs

Error Log

MySQL server logs any errors to the following logfile. Use this to troubleshoot the server.
\$DATA_DIR/\$hostname.err

Bin Logs

Binary logs are records of each UPDATE,DELETE,INSERT records that flows through the database. These can act as incremental backup files but are to be relied upon only as the last possible resort. They need to be converted from binary to sql format using the mysqlbinlog program. The files increment each time the server is restarted or once they reach the predefined limit as specified in the my.cnf file.

\$DATA_DIR/\$hostname-bin.xxxx

Replication

Theory and Requirements

Replication can take many forms. It can be used for:

- A server to run backups on so that the master server is not loaded with more process requests during essential hours.
- A server for READ-ONLY purposes to allow the application to spread database SELECT query load.
- A cluster of SLAVES to spread the SELECT load.
- A server for specific user needs, such as the marketing team that needs access to a specific table or view - then their reporting applications will not generate traffic on the master server which could impact OLTP performance, for example.

Replication Notes

- Each server in a master/slave situation must have a unique ID which is specified in the my.cnf file. Master is usually "1" and slaves in incrementing numbers.
- Slaves can only have one master server. This may change in 5.1 release.
- You cannot replicate data from a higher version of MySQL to a lower version of MySQL. If master server is 5.x and slave is 4.x then replication will not work.
- It is wise to keep user access accounts the same on master and slave unless you have specific reasons for doing so - such as the marketing example.
- You must have binary logging set for the specific hostname for each master/slave in the my.cnf file or replication will not function well, if at all.

Setup

First setup the replication user account on the master server. Verify the user was created by issuing the last statement.



Setting Up Replication

```
mysql> GRANT REPLICATION SLAVE ON . TO 'repl'@'%mydomain.com' IDENTIFIED BY
'slavepass';
mysql> FLUSH TABLES WITH READ LOCK;
mysql> FLUSH PRIVILEGES;
mysql> SELECT User,Host from mysql.user;
```

Then you need to copy the binary data files from the master host to the slave host as well as the my.cnf file. Be sure to change the hostname settings in the cnf file on the slave to reference the correct slave hostname. Also, add the authentication lines in the cnf file so that the slave can connect to the master to sync data.



Slave authentication my.cnf addition

```
server_id = unique_server_id
master_host=hostname_of_master
master_user='replication_user'
master_password='replication_password'
```

Get the master_log_file and master_log_pos and then copy the data over.



Copy Binary Data

```
MASTER: mysql>show master status;
```



master info

```
mysql>
mysql>show master status;
File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
lamp1-bin.000046 | 53804109 | | |
1 row in set (0.01 sec)
```

```
MASTER: #/etc/init.d/mysqld stop
SLAVE: #/etc/init.d/mysqld stop
SLAVE: mv $DATA_DIR /tmp/
SLAVE: rm -rf $DATA_DIR/*
```

```
MASTER: cd $DATA_DIR
scp -r * root@slave_server:$DATA_DIR/
```

```
SLAVE: verify that data has copied and start MySQL
#/etc/init.d/mysqld start
```

```
login to mysql
#mysql -uroot -p
mysql>stop slave;
mysql>change master to master_log_file='master_log_file', master_log_pos=master_log_pos;
```

```
MASTER: Start server
#/etc/init.d/mysqld start
```

```
SLAVE: Start replication sync
mysql>start slave;
mysql>show slave status\G
```



Example working status

```
mysql> show slave status\G
Slave_IO_State: Waiting for master to send event
Master_Host: 192.168.0.5
Master_User: replication
Master_Port: 3306
Connect_Retry: 60
Master_Log_File: lamp1-bin.000046
Read_Master_Log_Pos: 53730770
Relay_Log_File: lamp2-relay-bin.000056
Relay_Log_Pos: 25138617
Relay_Master_Log_File: lamp1-bin.000046
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Replicate_Do_DB:
Replicate_Ignore_DB:
Replicate_Do_Table:
Replicate_Ignore_Table:
Replicate_Wild_Do_Table:
Replicate_Wild_Ignore_Table:
Last_Errno: 0
Last_Error:
Skip_Counter: 0
Exec_Master_Log_Pos: 53730770
Relay_Log_Space: 42393543
Until_Condition: None
Until_Log_File:
Until_Log_Pos: 0
Master_SSL_Allowed: No
Master_SSL_CA_File:
Master_SSL_CA_Path:
Master_SSL_Cert:
Master_SSL_Cipher:
```

```
Master_SSL_Key:  
Seconds_Behind_Master: 0  
1 row in set (0.01 sec)
```

If there are errors, see troubleshooting below.

MASTER: Check to see if replication works
mysql>create database foo;

SLAVE: Check to see if database was replicated
mysql>show databases;

You should see the "foo" database

MASTER: Remove foo database.
mysql>drop database foo;

SLAVE: Check again.
mysql>show databases;

"foo" should not be listed

SLAVE: Remove old data contents
#rm -rf /tmp/\$DATA_DIR

Troubleshooting

Login to the server via the CLI.

Slave Status

```
mysql> show slave status\G  
Slave_IO_State: Waiting for master to send event  
Master_Host: 192.168.0.5  
Master_User: replication  
Master_Port: 3306  
Connect_Retry: 60  
Master_Log_File: lamp1-bin.000046  
Read_Master_Log_Pos: 11264544  
Relay_Log_File: lamp2-relay-bin.000054  
Relay_Log_Pos: 3056155  
Relay_Master_Log_File: lamp1-bin.000046  
Slave_IO_Running:
```



Check
[Yes/No]

Slave_SQL_Running:



Check

[Yes/No]

Replicate_Do_DB:
Replicate_Ignore_DB:
Replicate_Do_Table:
Replicate_Ignore_Table:
Replicate_Wild_Do_Table:
Replicate_Wild_Ignore_Table:
Last_Errno: 0
Last_Error:
Skip_Counter: 0
Exec_Master_Log_Pos: 11264544
Relay_Log_Space: 3056155
Until_Condition: None
Until_Log_File:
Until_Log_Pos: 0
Master_SSL_Allowed: No
Master_SSL_CA_File:
Master_SSL_CA_Path:
Master_SSL_Cert:
Master_SSL_Cipher:
Master_SSL_Key:
Seconds_Behind_Master:



Check

[0/NULL/X]

1 row in set (0.00 sec)

Any errors will need to be corrected.

1. If either SQL thread says NO or Seconds_Behind_Master=NULL or is a very high number then do the following. Use the "set global" command only if you have encountered a SQL error that has wedged replication. Otherwise just stop the slave and check the logs.



Sequence of Commands

```
mysql> stop slave;  
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> set global sql_slave_skip_counter=1;  
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> start slave;  
Query OK, 0 rows affected (0.00 sec)
```

Check status again

Run the "show slave status\G" command again and see if the threads are set to "YES" and the ERROR has cleared, as well as a decreasing "seconds_behind_master" number if applicable. Run the status command several times to make sure replication continues to work.

If these commands do not bring replication back online then consult the error logs and dig deeper into your troubleshooting abilities. Check availability of network, binary log files, and any pertinent information given in the error log. This document cannot scope every single error but the previous are the most common solutions.

Clustering

Theory and Requirements

Clustering requires multiple nodes, the number depending on the setup and methods used to manage data. Clustering scales more efficiently than replication and does not require application specific logic to read from one IP and write to another IP. Clustering can require more or less overall maintenance than replication depending on the methods used to implement.

Clustering Methods

MySQL Cluster via NDBCLUSTER database engine

Does not have foreign key support.
Limited to Read-Committed locking contention.
Requires minimum of 3 nodes - 2x sql/data nodes, 1 management node. Scales very well.
Uses heartbeat and Idirector to initiate VIP IP and manage traffic flow.
Requires cross-connects on non-routable subnet to manage heartbeat.
Preferred method of clustering if user does not need InnoDB engine features.
Require low latency network

Redhat Cluster Services w/ SAN (not covered by this version of the document)

Can use InnoDB engine for transaction and Serializable locking contention.
Requires SAN storage for shared data mount.
Requires Redhat license for the cluster management software.
Requires cross-connects on non-routable subnet to manage heartbeat.
Require low latency network

DRBD for local disk to local disk sync (not covered by this version of the document)

Uses heartbeat and Idirector to initiate VIP IP and manage traffic flow.
Does not need shared data mount, DRBD syncs data between each node's local disks.
Requires cross-connects on non-routable subnet to manage heartbeat.
Require low latency network

NDB Clustering differs from replication in several ways.

Uses multiple hosts to share the load via a VIP IP Address.

Nodes can be added to increase performance as growth or traffic needs require.
All nodes share the same data at the same time, there is no latency or asynchronous transfer from node to node as there is in replication setups.
Share nothing architecture allows that if one node fails no other nodes are affected in regard to data or availability.

Setup

Download the following RPMs for your servers.

MySQL-bench-5.0.21-0.glibc23.i386.rpm MySQL-server-5.0.22-0.glibc23.i386.rpm
MySQL-client-5.0.21-0.glibc23.i386.rpm MySQL-shared-5.0.21-0.glibc23.i386.rpm
MySQL-devel-5.0.21-0.glibc23.i386.rpm MySQL-shared-compat-5.0.21-0.glibc23.i386.rpm
MySQL-Max-5.0.21-0.glibc23.i386.rpm
MySQL-ndb-extra-5.0.21-0.glibc23.i386.rpm
MySQL-ndb-management-5.0.21-0.glibc23.i386.rpm
MySQL-ndb-storage-5.0.21-0.glibc23.i386.rpm
MySQL-ndb-tools-5.0.21-0.glibc23.i386.rpm

Alternately download the attached tar.gz file for automated clustering installation.

This includes the RPMs and custom installation scripts I have written to setup load-balancing, server install, management node install, and kernel modules. Also download the "LoadBalancing-Configs.txt" file for settings used in the setup of the cluster.

For the most detailed information of setting up MySQL Cluster please see the book entitled: "MySQL Clustering" from MySQL Press. No need to reinvent the wheel.

Troubleshooting

Placeholder.

Backups

Scripts and GUI

The "mysqldump" utility is the primary tool used to backup MySQL databases. Please consult the many options for the tool to properly use it for a give database. Recommended is the "--routines" always be used in the case that customers are using SPs.

I have written a GUI wrapper webapp for this which can be found here
<http://sourceforge.net/projects/monolith-mysql/>

Security

User Privileges

Grant users access only to databases they need access to

i all access
mysql>grant all on database.* to 'username'@'host' identified by 'password';

OR

i limited access
mysql>grant select,update,"etc" on database.* to 'username'@'host' identified by 'password';

Host Privileges

Only allow user access from trusted hosts, localhost. Wildcards of "%" can be used in the appropriate octet to allow connections from various subnets/supernets.

i limited networks
mysql>... 'username'@'localhost'... **or** mysql>... 'username'@'192.168.0.4'... **or** mysql>... 'username'@'192.168.0.%'...

If you absolutely need to, or are in a network that allows such open security you can grant the user as such:

i all networks
mysql>..... 'username'@'%' ...

Access and Authentication

Socket vs. Port

Connections made to the MySQL server on the localhost will connect to the open socket as specified in the my.cnf file. You can specify a socket file as such:

i socket connections
#mysql --user=username -p --socket=/var/lib/mysql/mysql.sock

i port connections
#mysql --user=username -p --port=3306

Methods

- CLI - the preferred manner for competent DBAs
- MySQL GUI Tools - useful for editing SPs, functions, triggers
- PhpMyAdmin - useful when you do not have shell or non-localhost access

