ALTNIX Research & Development Team

ALTNIX Research & Development Team
□ Mr Sadhiq Basheer Ahmed
□ Mr Rahul Walunj
□ Mr Rahul Patil
□ Mr Venood khatuva
□ Mr Javed Shaikh
□ Mr Avinash Kumar Ojha

[In this document you will learn all about MySQL admin related work, like MySQL backup, replication, MySQL Clusters etc.]

About MySQL:

MySQL is used in almost all the open source web projects that require a database in the back-end. MySQL is part of the powerful LAMP stack along with Linux, Apache and PHP. this was originally created by a company called MySQL AB, which was acquired by Sun Microsystems, which was acquired by Oracle. Since we don't what Oracle will do with MySQL database, open source community has created several forks of MySQL including Drizzle and MariaDB.

Table of Contents

Storage Engines

MyISAM manages non transactional tables. It provides high-speed storage and retrieval, as well as fulltext searching capabilities. MyISAM is supported in all MySQL configurations, and is the default storage engine unless you have configured MySQL to use a different one by default for until Version 5.4 From Version 5.5 default engine is InnoDB.

The 2 major types of table storage engines for MySQL databases are InnoDB and MyISAM. To summarize the differences of features and performance.

The MEMORY storage engine provides in-memory tables. The MERGE storage engine enables a collection of identical MyISAM tables to be handled as a single table. Like MyISAM, the MEMORY and MERGE storage engines handle non transactional tables, and both are also included in MySQL by default.

The InnoDB and BDB storage engines provide transaction-safe tables. To maintain data integrity, InnoDB also supports FOREIGN KEY referential-integrity constraints.

MyISAM vs Innodb - Quick comparison Table:

	INODDB	MYISAM
1	It's New Engine	It's Old Engine
2	InnoDB is more complex	MYISAM is Simpler
3	InnoDB is more strict in data integrity	MYISAM is loose in data integrity
4	InnoDB implements row-level lock for inserting and updating	MyISAM implements table-level lock.
5	InnoDB has foreign keys and relationship contraints	MyISAM does not.
6	InnoDB has better crash recovery	MyISAM is poor at recovering data integrity at system crashes.
7	InnoDB does not have full-text search index.	MyISAM has full-text search index
8	Not *ACID compliant and non-transactional	*ACID compliant and hence fully transactional with ROLLBACK and COMMIT and support for Foreign Keys

Advantages and Disadvantages

In light of these differences, InnoDB and MyISAM have their unique advantages and disadvantages against each other. They each are more suitable in some scenarios than the other

Advantages of InnoDB

1. InnoDB should be used where data integrity comes a priority because it inherently takes care of them by the help of relationship constraints and transactions.

2. Faster in write-intensive (inserts, updates) tables because it utilizes row-level locking and only hold up changes to the same row that's being inserted or updated.

Disadvantages of InnoDB

1. Because InnoDB has to take care of the different relationships between tables, database administrator and scheme creators have to take more time in designing the data models which are more complex than those of MyISAM.

2. Consumes more system resources such as RAM. As a matter of fact, it is recommended by many that InnoDB engine be turned off if there's no substantial need for it after installation of MySQL.

3. No full-text indexing.

Disadvantages of MyISAM

1. No data integrity (e.g. relationship constraints) check, which then comes a responsibility and overhead of the database administrators and application developers.

2. Doesn't support transactions which are essential in critical data applications such as that of banking.

3. Slower than InnoDB for tables those are frequently being inserted to or updated, because the entire table is locked for any insert or update.

The comparison is pretty straightforward. InnoDB is more suitable for data critical situations that require frequent inserts and updates. MyISAM, on the other hand, performs better with applications that don't quite depend on the data integrity and mostly just select and display the data.

TABLES AND COLUMNS

Limits on Table Column Count and Row Size

There is a hard limit of 4096 columns per table, but the effective maximum may be less for a given table. The exact limit depends on several interacting factors.

Every table (regardless of storage engine) has a maximum row size of 65,535 bytes. Storage engines may place additional constraints on this limit, reducing the effective maximum row size.

The maximum row size constrains the number (and possibly size) of columns because the total length of all columns cannot exceed this size. For example, utf8 characters require up to three bytes per character, so for a CHAR(255) CHARACTER SET utf8 column, the server must allocate 255 × 3 = 765 bytes per value. Consequently, a table cannot contain more than 65,535 / 765 = 85 such columns.

TYPES OF STRINGS

String Types
1.1. The CHAR and VARCHAR Types
1.2. The BINARY and VARBINARY Types
1.3. The BLOB and TEXT Types
1.4. The ENUM Type
1.5. The SET Type

CHAR

The length of a CHAR column is fixed to the length that you declare when you create the table. The length can be any value from 0 to 255. When CHAR values are stored, they are right-padded with spaces to the specified length. When CHAR values are retrieved, trailing spaces are removed unless the PAD_CHAR_TO_FULL_LENGTH SQL mode is enabled.

VARCHAR

Values in VARCHAR columns are variable-length strings. The length can be specified as a value from 0 to 65,535. The effective maximum length of a VARCHAR is subject to the maximum row size (65,535 bytes, which is shared among all columns) and the character set used. See Section E.10.4, “Limits on Table Column Count and Row Size”.

BINARY and VARBINARY

The BINARY and VARBINARY types are similar to CHAR and VARCHAR, except that they contain binary strings rather than nonbinary strings. That is, they contain byte strings rather than character strings. This means that they have no character set, and sorting and comparison are based on the numeric values of the bytes in the values.

BLOB

A BLOB is a binary large object that can hold a variable amount of data. The four BLOB types are TINYBLOB, BLOB, MEDIUMBLOB, and LONGBLOB. These differ only in the maximum length of the values they can hold. The four TEXT types are TINYTEXT, TEXT, MEDIUMTEXT, and LONGTEXT. These correspond to the four BLOB types and have the same maximum lengths and storage requirements. See Section 11.6, “Data Type Storage Requirements”.

DIFF CHAR VS VARCHAR

In general, varchar(255) requires as much storage as varchar(1). In each case the table stores something like a pointer into a string table and a length. E.g. 4 bytes offset + 1 byte size = 5 bytes fixed per row, just for overhead.

The actual content is of course in the string table, which is only as long as the string your store in it. So if you store a 5 letter name in a varchar(255) field, it'll only use (say) 5 overhead bytes + 5 content bytes = 10 bytes.

Using a varchar(10) field will use exactly the same amount, but will only truncate strings longer than 10 bytes.

A varchar won't take up more space than the string you store in it, aside from the overhead for storing the string length:

Value	CHAR(4)	Storage Required	VARCHAR(4)	Storage Required
''	' '	4 bytes	''	1 byte
'ab'	'ab '	4 bytes	'ab'	3 bytes
'abcd'	'abcd'	4 bytes	'abcd'	5 bytes
'abcdefgh'	'abcd'	4 bytes	'abcd'	5 bytes

s

However, if you really do only require 5 chars, then consider using char(5) if there are no other variable width columns in the table (i.e., varchars, text or blobs). Then you will have fixed length record, which does carry some performance advantages:

For MyISAM tables that change frequently, you should try to avoid all variable-length columns (VARCHAR, BLOB, and TEXT). The table uses dynamic row format if it includes even a single variable-length column.

Field indicates the column name.

Type indicates the column data type.

NULL

The Null field contains YES if NULL values can be stored in the column. If not, the column contains NO as of MySQL 5.0.3, and " before that.

The NULL value can be surprising until you get used to it. Conceptually, NULL means “a missing unknown value” and it is treated somewhat differently from other values.

In MySQL, 0 or NULL means false and anything else means true. The default truth value from a boolean operation is 1.

Key

The Key field indicates whether the column is indexed:

If Key is empty, the column either is not indexed or is indexed only as a secondary column in a multiple-column, nonunique index.

Extra

The Extra field contains any additional information that is available about a given column. The value is auto_increment for columns that have the AUTO_INCREMENT attribute and empty otherwise.

Privileges indicates the privileges you have for the column. This value is displayed only if you use the FULL keyword.

MySQL Shutdown Safely
Reset MySQL root password

$ service mysqld stop

$ mysqld_safe --skip-grant-tables &

mysql_safe is recoommended way to start a mysql server on Unix and NetWare.
mysql_safe adds some safety features such as restarting the server when an error occurs and logging runtime information to an error log file

```
mysql> update user set password=PASSWORD("test123") where User='root';
```

Once you have given the desired privileges for your user, you will need to FLUSH privileges in order to complete the setup and to make the new settings work

```
$ service mysqld stop
$ service mysqld start
```

To check updated tables

```
$ myisamchk /var/lib/mysql/mysql/db.MYI
```

To Create database with management tool

```
$ mysqladmin -h localhost -u root -p'YourPassword' create bedrock
```

OR using

```
mysql> create database bedrock;
```

Specify database to connect to. Also refers to path: /var/lib/mysql/bedrock

```
mysql> use bedrock;
```

```
mysql> create table employee (Name char(20),Dept char(20),jobTitle char(20));
```

View table just created. Same as "show columns from employee;"

```
mysql> DESCRIBE employee;
```

Output

```
+----------+----------+------+-----+---------+-------+
| Field    | Type     | Null | Key | Default | Extra |
+----------+----------+------+-----+---------+-------+
| Name     | char(20) | YES  |     | NULL    |       |
| Dept     | char(20) | YES  |     | NULL    |       |
| jobTitle | char(20) | YES  |     | NULL    |       |
+----------+----------+------+-----+---------+-------+
3 rows in set (0.03 sec)
```

```
mysql> show tables;
```

Output

```
+------------------+
| Tables_in_bedrock |
+------------------+
| employee         |
+------------------+
```

```
mysql> INSERT INTO employee VALUES ('Dhoni','ITDept','Sr. Linux Admin');
```

```
mysql> INSERT INTO employee VALUES ('Rahul','Devlopment','Sr. Devloper');
```

To View Table :

```
mysql> select * from employee;
```

Output

```
+----------------+---------------+-----------------+
| Name           | Dept          | jobTitle        |
+----------------+---------------+-----------------+
| Dhoni          | ITDept        | Sr. Linux Admin |
| Sachin         | Devlopment    | Sr. Devloper    |
+----------------+---------------+-----------------+
3 rows in set (0.00 sec)
```

Note: Data type used was CHAR. Other data types include:

CHAR(M) : Fixed length string. Always stores M characters whether it is holding 2 or 20 characters. Where M can range 1 to 255 characters.

VARCHAR(M) : Variable length. Stores only the string. If M is defined to be 200 but the string is 20 characters long, only 20 characters are stored. Slower than CHAR.

INT : Ranging from -2147483648 to 2147483647 or unsigned 0 to 4294967295

FLOAT(M,N) : FLOAT(4,2) - Four digits total of which 2 are after the decimal. i.e. 12.34 Values are rounded to fit format if they are too large.
DATE, TEXT, BLOB, SET, ENUM

How to Create user in Mysql

```
mysql -h localhost -u root -p'password'
```

```
mysql> USE mysql;
```

```
mysql> SHOW TABLES;
+-----------------+
| Tables_in_mysql |
+-----------------+
| columns_priv    |
| db              |
| func            |
| host            |
| tables_priv     |
| user            |
+-----------------+
```

```
mysql> INSERT INTO user (Host, User, Password, Select_priv) VALUES ('', 'Dude1', password('supersecret'), 'Y');
```

Required each time one makes a change to the GRANT table

```
mysql> FLUSH PRIVILEGES;
```

```
$mysql> GRANT ALL PRIVILEGES ON bedrock.* TO Dude1;
```

```
mysql> FLUSH PRIVILEGES;
mysql> quit
```

Test the database:

```
mysql> use bedrock;
```

```
mysql> SELECT name FROM employee WHERE dept='ITDept';
```

OutPut:

```
+-------+
| name  |
+-------+
| Dhoni |
+-------+
1 row in set (0.00 sec)
```

Security

Security and database access is controlled by the GRANT tables. Access to connect to the database and access to process the transaction (table and column access, etc.) are both required. Privileges are searched in the following order:

1. user table
2. db and host table
3. tables_priv
4. columns_priv

His grants access from nodes 192.168.10.0 - 192.168.10.255. Or the network definitions can reference resolvable names: '%.domain.com'. The host definition of '%' or '' (null) refers to any host. (..according to the documentation. My experience is that in the mysql.user table use only '%' for "Host" to refer to any host.)

```
mysql> GRANT ALL PRIVILEGES on bedrock.* to david@'%';

mysql> FLUSH PRIVILEGES;
```

or (more promiscuous)

```
mysql> GRANT ALL PRIVILEGES on *.* to david@'%' identified by 'david';
mysql> FLUSH PRIVILEGES;
```

```
mysql> SHOW GRANTS FOR david@'%';
```

Output:

```
+-------------------------------------------------+
| Grants for david@%                              |
+-------------------------------------------------+
| GRANT USAGE ON *.* TO 'david'@'%'               |
| GRANT ALL PRIVILEGES ON `bedrock`.* TO 'david'@'%' |
+-------------------------------------------------+
2 rows in set (0.00 sec)
```

How to Delete User from mysql

```
mysql> USE mysql;
```

```
mysql> SELECT User,Password,Host from user;
```

OutPut:

```
+-------+------------------+------------------+
| User  | Password         | Host             |
+-------+------------------+------------------+
| root  | 39817a786ddf7333 | localhost        |
| root  | 39817a786ddf7333 | monit.home.local |
| root  | 39817a786ddf7333 | 127.0.0.1        |
|       |                  | localhost        |
|       |                  | monit.home.local |
| david |                  | %                |
+-------+------------------+------------------+
7 rows in set (0.00 sec)
```

```
mysql> DELETE FROM user WHERE User='david' AND Host='localhost';
```

Now check

```
mysql> SELECT User,Password,Host from user;
```

```
mysql> FLUSH PRIVILEGES;

mysql> QUIT
```

Any changes (UPDATE) to the user table will require a "FLUSH PRIVILEGES" before the changes will be effective.

```
mysql> UPDATE user SET Host='%' WHERE User='Dude2';
mysql> FLUSH PRIVILEGES;
```

To Drop/Delete Particular Database

```
mysql> drop database bedrock;
```

OR

```
mysqladmin -u root -p drop database-name
```

Show list of active threads in server

```
mysqladmin -h localhost -u root -p processlist
```

OutPut:

```
+----+------+-----------+-------+---------+------+-------+------------------+
| Id | User | Host      | db    | Command | Time | State | Info             |
+----+------+-----------+-------+---------+------+-------+------------------+
| 18 | root | localhost | cacti | Sleep   | 1    |       |                  |
| 20 | root | localhost |       | Query   | 0    |       | show processlist |
+----+------+-----------+-------+---------+------+-------+------------------+
```

We can use comment in SQL Queries followed by – (double dash) sign

Example:

```
mysql> CREATE DATABASE bedrock;  -- Comments follow a double dash
```

Interrogate an existing database:
in Below Example You will learn :

- How to use Regular expression to search any data
- How to Use AND and OR in Sql Queries to match specific condition

```
mysql> USE bedrock;

$mysql> SELECT DATABASE();  -- returns current database. eg. bedrock

$mysql> SELECT VERSION();

$mysql> SELECT NOW();

$mysql> SELECT USER();

$mysql> SHOW TABLES;

$mysql> DESC employee;

$mysql> SHOW CREATE TABLE employee;  -- show command used to generate table

$mysql> SHOW INDEX FROM employee;

$mysql> SELECT DISTINCT dept FROM  bedrock;

$mysql> SELECT * FROM  bedrock WHERE Name  LIKE "B%y";  -- "%" match any char: Gives Betty and Barney

$mysql> SELECT * FROM  bedrock WHERE Name LIKE "B_ _y";  -- "_" match space: Gives Betty but not Barney
```

```
$mysql> SELECT * FROM  bedrock WHERE Name RLIKE "^Betty$";  -- "^" match beginning. "$" to denote end of string
$mysql> SELECT COUNT(*) FROM employee;  -- Number of records returned
$mysql> SELECT Name, COUNT(*) FROM employee WHERE Name LIKE "B%y";  -- Return Names and number of records returned
$mysql> SELECT * FROM pet WHERE species = "snake" OR species = "bird";
$mysql> SELECT * FROM pet WHERE species = "dog" AND sex = "f";
$mysql> SELECT * FROM pet WHERE birth >= "1998-1-1";
$mysql> SELECT name, birth FROM pet ORDER BY birth DESC;
$mysql> SELECT * FROM pet WHERE name LIKE "b%";
$mysql> SELECT * FROM pet WHERE name REGEXP "^b";
$mysql> SELECT species, COUNT(*) FROM pet GROUP BY species;
$mysql> SELECT MAX(article) AS article FROM shop;
$mysql> SELECT * FROM employee WHERE name LIKE "%Sm%";
$mysql> SELECT * FROM employee WHERE name REGEXP "^Ja";
```

Database cleanup:

```
mysql> DROP TABLE employee;
mysql> DROP DATABASE bedrock;
mysql> DROP DATABASE database_name;
```

The DROP DATABASE statement removes from the given database directory those files and directories that MySQL itself may create during normal operation:

- All files with the following extensions.

.BAK	.DAT	.HSH	.MRG
.MYD	.MYI	.TRG	.TRN
.db	.frm	.ibd	.ndb

Mysql data

<TableName>.frm

This file is key to both InnoDB and MyISAM databases. It is the Meta data to the location of your data. It contains the table column definitions.

If you remove this file MySQL will tell you your DATA doesn't exist. It does. Your data is still in the ibdata (.ibd) file or the ibdata1 file. You need to recreate the table to recreate this file.

If you don't know the exact structure of this table you're out of luck.

Stop MySQL and move the .MYD and .MYI files to another directory. (You might also make a backup copy.) Start MySQL and recreate this table. Stop MySQL and copy the .MYD and .MYI files back to the database directory and restart MySQL.

<TableName>.MYD

THIS IS YOUR MyISAM DATA. If this is all you have, and you know the data structure of the the table, all is not lost. (See .frm Above.) You may also need to recreate the .MYI index file.

<TableName>.MYI

This file contains the indexs for your table.
If it becomes corrupt or is deleted you can recreate it using the '

REPAIR TABLE table_name USE_FRM;'

command.

/tmp/#<TableName>.[frm MYD MYI]

http://nilinfobin.com/2012/01/some-facts-about-myisam-file-handling-and-myi-file-in-mysql/

MySQL creates temporary files in the /tmp director when sorting (ORDER BY or GROUP BY).
It also creates temp files on a replication server when the LOAD DATA LOCAL INFILE… command is used.
Replication will fail if these files are removed.

<TableName>.ibd

THIS IS YOUR InnoDB DATA. If this is all you have, and you know the data structure of the the table, all is not lost. (See .from Above.)
Unlike MyISAM tables the indexes are contained in this file with your data.

MySQL doesn't create this file unless you are using the innodb_file_per_table option.

If you have a "Clean" .ibd file you can import it into a database with the command

```
mysql> ALTER TABLE tbl_name IMPORT TABLESPACE;
```

ib_logfile*

This file contains your un-committed transactions data. MySQL uses it to recover from a crash.

If you shut down InnoDB cleanly, you can remove them. MySQL will recreate them.

If you change the size of innodb_log_file_size, you will need to recreate these files by stopping MySQL cleanly and deleting them.

IBDATA

ibdata1 is the data file for tables stored using the InnoDB engine.

The ibdata1 file is part of the InnoDB engine in MySQL. It is a very important file as part of the Microsoft program.

If the file is deleted you will not be able to get MySQL to restart. If you do not work with the InnoDB engine you may be able to remove the file safely as long as you delete the ib_logfile* files too.

MySQL uses these for reference for the ibdata 1 file. The program will not work if the ib_logfile* is still there since it is a reference. It feels there is something wrong with the software program and the references

InnoDB does not create directories, so make sure that the /ibdata directory exists before you start the serve

Two important disk-based resources managed by the InnoDB storage engine are its tablespace data files and its log files. If you specify no InnoDB configuration options, MySQL creates an auto-extending 10MB data file named ibdata1 and two 5MB log files named ib_logfile0 and ib_logfile1 in the MySQL data directory. To get good performance

1.This setting configures a single 10MB data file named ibdata1 that is auto-extending. No location for the file is given, so by default, InnoDB creates it in the MySQL data directory.

Sizes are specified using K, M, or G suffix letters to indicate units of KB, MB, or GB.

A tablespace containing a fixed-size 50MB data file named ibdata1 and a 50MB auto-extending file named ibdata2 in the data directory can be configured like this:

```
[mysqld]
innodb_data_file_path=ibdata1:50M;ibdata2:50M:autoextend
```

Set max

```
[mysqld]
innodb_data_file_path=ibdata1:10M:autoextend:max:500M
```

Set buffer pool size to 50-80% of your computer's memory,

but make sure on Linux x86 total memory usage is < 2GB

Mysql Inoddb

MyISAM tables yield better performance results in selecting data records

InnoDB tables give better performance results in inserting and updating data records

To change a MyISAM table to an InnoDB table:

```
msyql> ALTER TABLE table_name ENGINE = InnoDB;
```

To convert an InnoDB table to a MyISAM table:

```
$msyql> ALTER TABLE table_name ENGINE = MyISAM;
```

Enable Inoddb

1. **innodb_data_file_path = ibtadta1:100M:autoextend**
2. **innodb_buffer_pool_size=500M**
3. **innodb_additional_mem_pool_size=100M**
4. **innodb_log_file_size=200M**
5. **innodb_log_buffer_size=80M**
6. **innodb_flush_log_at_trx_commit=1**

MYSQL BACKUP TOOLS

MYSQLDUMP

MYSQLHOTCPY

Configuring MYSQL REPLICATION
mdraid1

- **Linux softraid level 1. If you run MySQL on mdraid1, just plug in another disk, let it sync, stop mysql, do a sync, remove the third and synced raid component, restart mysql. If you use bitmaps with your raid1, a re-sync to take another snapshot is usually quite fast....**
-

MySQL in a sandbox

WordPress plugin
(WP-DB-Backup)
This plugin can backup the database on a schedule and email it to you(assuming its a small database). You can also just back it up locally.

XtraBackup from Percona.
XtraBackup **is a comprehensive tool that behaves like rsync but has directed purpose in life. It can start off copying all InnoDB data and tablespaces. It has the ability to create checkpoints internally and perform in-place an InnoDB crash recovery to help get perfect point-in-time backup. XtraBackup also has an extra feature which allows for the creation of incremental backups.**

Automysqlbackup hell script
http://sourceforge.net/projects/automysqlbackup/**.**

MAATKIT

CDP R1Soft **(**MySQL Module Option **that takes point-in-time snapshots)**

MySQL Enterprise Backup **(formerly InnoDB Hot Backups [commercial])**

LVMSNAPSHOT

mysqldump is an effective tool to backup MySQL database. It creates a *.sql file with DROP table, CREATE table and INSERT into sql-statements of the source database. To restore the database, execute the *.sql file on destination database. For MyISAM, use mysqlhotcopy method **that we explained earlier, as it is faster for MyISAM tables.**

http://www.thegeekstuff.com/2008/09/backup-and-restore-mysql-database-using-mysqldump/

mysqldump

```
mysqldump -u[USERNAME] -p[PASSWORD] --add-drop-table --no-data [DATABASE] | \
grep ^DROP | mysql -u[USERNAME] -p[PASSWORD] [DATABASE]
```

Backup

```
mysqldump -h YOUR.DBSERVER.COM -uUSERNAME -pPASSWORD DATABASENAME > \
$HOME/html/backups/database/$NOWDATE.sql
```

Compressed backup file

```
gzip -9 $HOME/html/backups/database/$NOWDATE.sql
```

Reference Link

http://www.noupe.com/how-tos/10-ways-to-automatically-manually-backup-mysql-database.html

http://www.ducea.com/2007/07/25/dumping-mysql-stored-procedures-functions-and-triggers/

Mysqldump Slow

If mysqldump performing slow then you can refer below link

http://dba.stackexchange.com/questions/4628/users-complain-that-system-runs-slow-when-mysqldump-is-in-progress

MYSQLHOTCOPY

mysqlhotcopy is a perl script that comes with MySQL installation. This locks the table, flush the table and then performs a copy of the database. You can also use the mysqlhotcopy to automatically copy the backup directly to another server using scp

How do I repair a corrupted or damaged table?

Tables, especially on busy sites can often be corrucpted. Serv crashes, or any other unexpected termination of a process before it's completed can and does corrupt tables. When this happens, it's nice to know the following can quickly repair the damage.

```
mysql> REPAIR TABLE tablename
```

drop table will remove the entire table with data

```
mysql> delete * from table
```

will remove the data, leaving the autoincrement values alone. it also takes a while if there's a lot of data in the table.

```
mysql> truncate table;
```

will remove the data, reset the autoincrement values (but leave them as autoincrement columns, so it'll just start at 1 and go up from there again), and is very quick.

Drop - deletes the data as well as structure

Truncate - deletes only the data, and resets the auto increment column to 0

Delete - Deletes the selected/all rows from a table, it doest not reset auto increment., here we can delete set of records by specifying the condition in where clause

Password recovery and change root password

http://www.cyberciti.biz/tips/recover-mysql-root-password.html

http://www.cyberciti.biz/faq/mysql-change-root-password/

Running Multiple Instance

http://www.ducea.com/2009/01/19/running-multiple-instances-of-mysql-on-the-same-machine/

Mysql slow

http://www.ducea.com/2006/11/06/identifying-mysql-slow-queries/

http://www.mysqlperformanceblog.com/2006/06/09/why-mysql-could-be-slow-with-large-tables/

Mysqldump Slow

http://dba.stackexchange.com/questions/4628/users-complain-that-system-runs-slow-when-mysqldump-is-in-progress

Mysql security

http://www.greensql.com/articles/mysql-security-best-practices

http://dev.mysql.com/doc/refman/5.0/en/default-privileges.html

Remove wildcards in the grant tables

Require the use of secure passwords

Check the permissions of configuration files

Encrypt client-server transmissions

Disable remote access

Actively monitor the MySQL access log

Mysql Interview

http://www.techinterviews.com/29-mysql-interview-questions

REFRENCE

http://www.cyberciti.biz/tips/how-do-i-enable-remote-access-to-mysql-database-server.html

http://www.cyberciti.biz/tips/howto-copy-mysql-database-remote-server.html

MYSQLCHECK

http://www.thegeekstuff.com/2011/12/mysqlcheck/

MysqlADMIN COMMANDS

http://www.thegeekstuff.com/2009/01/15-practical-usages-of-mysqladmin-command-for-administering-mysql-server/

http://www.ramblingsofasysadmin.com/2006/09/showing_and_understanding_mysq.html

http://www.thegeekstuff.com/2011/10/mysql-tutorial-basics/

http://www.thegeekstuff.com/2008/07/howto-install-mysql-on-linux/

http://www.thegeekstuff.com/2011/12/mysqlcheck/

http://www.thegeekstuff.com/2009/01/15-practical-usages-of-mysqladmin-command-for-administering-mysql-server/

http://www.yolinux.com/TUTORIALS/LinuxTutorialMySQL.html

MYSQL Replication from **One Master** Server & **Two Slave** Server

Step on **Master Server**

Step 1# Install MySQL Server Packages and Configure **/etc/my.cnf** as below

```
[mysqld]
datadir = /var/lib/mysql
socket = /var/lib/mysql/mysql.sock

log-bin = mysql-bin  #----> imp for replication

server-id = 1  # ---> imp for replication

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

In above conf file **server id = 1** which indicate **MASTER Server**
Login SQL Server by **mysql -u root -p**

Step 2# Permission

```
mysql> show databases;

mysql> grant replication slave on *.* to repl@'172.16.1.10' identified by 'hello123';
```

If you face any error then run the following

```
mysql> grant all priviliges on *.* to repl@'172.16.1.10' identified by 'hello123';
```

The above will do important role in /etc/my.cnf file of Slave SQL Server

Step 3# Take existing database dump

```
mysqldump -u root -p --databases  database_name  > dbdump.sql
```

The above command will create a particular database dump in sql format into a present path on which command was run

Step 4# Now copy this dbdump.sql in slave server by scp

Step on **Slave Server**

1. Install mysql & configure /etc/my.cnf file is as follows
2. mv dbdump.sql in /var/lib/mysql/dbdump.sql
3. Restore the back up by

```
mysql -u root -p < /path/of/dump/file/dbdump.sql
```

4. vim /etc/my.cnf

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
#user=mysql
server-id=2                 # server-id=3
master_host = 172.16.1.10       #   (For 2nd Slave Server)
master_user = repl
master_password = hello123
master_port = 3306
log-bin = mysql-bin
# Default to using old password format for compatibility with mysql 3.x
# clients (those using the mysqlclient10 compatibility package).
#old_passwords=1
#relay-log = mysql-relay-bin
[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

5. In above conf file **server-id=2** which indicate **SLAVE Server**,

master_host = 172.16.1.10 □ Master Server
master_user = repl □ Authenticated user
master_password = hello123 □ Password
master_port = 3306 □ Port No.
(The above will compare with Master Server

mysql> grant replication slave on *.* to repl@'172.16.1.10' identified by 'hello123';)

On Slave Server

1.

```
mysql> show slave status \G;   --- look for Seconds_Behind_Master
```

Seconds_Behind_Master: 0 >> Master & Slave Server Communication is OK

Seconds_Behind_Master: NULL >> Need to do following step to perform OK

2.

```
mysql> stop slave;
```

```
mysql> change master to master_host='172.16.1.10', master_user='repl', master_password='hello123',
master_log_file='mysql-bin.000113',     master_log_pos=946;
```

In above command values of '**master_log_file**' & '**master_log_pos**' will came
From Master Server by giving the command
mysql> show master status;
and take out put of the **'File'** & **'Position'**

3.

```
mysql> start slave;
```

And Again look mysql> show slave status \G; for **Seconds_Behind_Master: 0**

Now create database on Master Server which will instantly get replicate on
Slave Server1 & Slave Server2

Only Server ID is Different of two Slave Server otherwise step & conf file is the same.

Mysql Replication working
http://crazytoon.com/2008/04/17/mysql-replication-series-how-does-mysql-replication-works/

http://crazytoon.com/2008/04/21/mysql-replication-replicate-by-choice/

Integrate Apache with MySQL for Authenticate Clients

Install Module

```
yum install mod_auth_mysql
```

Create User Database

```
mysql -u root -p
mysql> create database auth;
mysql> use auth;
mysql> grant all on auth.* to auth_user@localhost identified by 'test123';
mysql> flush privileges;
mysql> CREATE TABLE users (
user_name CHAR(30) NOT NULL,
user_passwd CHAR(20) NOT NULL,
PRIMARY KEY (user_name)
);
mysql> CREATE TABLE groups (
user_name CHAR(30) NOT NULL,
user_group CHAR(20) NOT NULL,
PRIMARY KEY (user_name, user_group)
);
```

Add users to MySql database

```
mysql> INSERT INTO users VALUES ('ram', 'test');
```

Open “/etc/httpd/conf.d/mod_auth_mysql.conf" with your favourite editor and do following changes

```
<VirtualHost *:80>
   <Directory "/var/www/html" >
      AuthName "MySQL authenticated zone"
      AuthType Basic
      AuthUserFile /dev/null
      AuthBasicAuthoritative Off
      AuthMySQLEnable On
      AuthMySQLAuthoritative On
      AuthMySQLHost 127.0.0.1
      AuthMySQLDB auth
      AuthMySQLUser auth_user
      AuthMySQLPassword test123
      AuthMySQLUserTable users
      AuthMySQLNameField user_name
      AuthMySQLPasswordField user_passwd
      AuthMySQLGroupField user_group
      # AuthMySQLPwEncryption md5
      AuthMySQLPwEncryption  none
      require valid-user
   </Directory>
</VirtualHost>
```

Note:- above parameter only applicable for CentOs/RHEL 5

Now Just Restart Apache Service and Login using user **ram**

Building database clusters with MySQL

MySQL is a mainstay of many web based applications, and is popular with lots of our customers. There does comes a time when a single database server is not enough. To enhance redundancy MySQL has a couple of options. You can add more servers with vanilla replication enabled. Or you can look at setting up MySQL Cluster. The documentation has this to say...

MySQL Cluster is designed not to have any single point of failure.

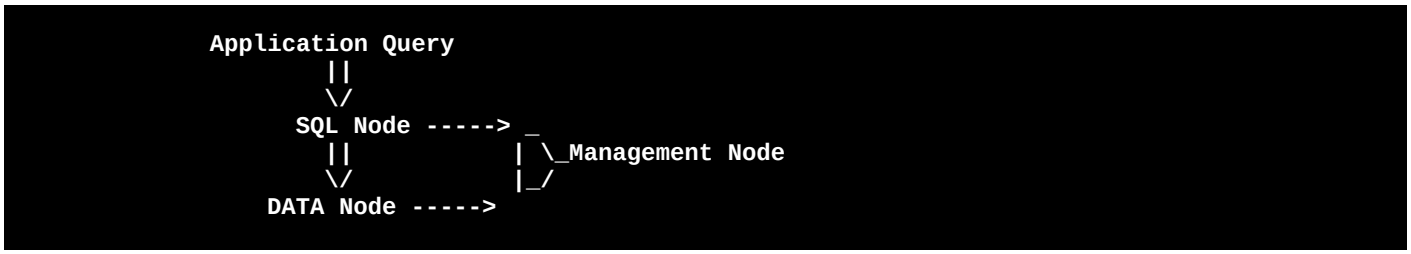
Setup Overview

Server Details:

*management node: hostname: mgr,IP: 192.168.2.53, tcp port: 2205

* sql node: hostname: sql1, private IP: 192.168.2.60
* sql node: hostname: sql2, private IP: 192.168.2.11

* data node: hostname: ndb1, private IP: 192.168.2.11
* data node: hostname: ndb2, private IP: 192.168.2.60



Logon to management node server mgr1 as root using an ssh session and download the latest mysql cluster installer with the following commands, as of this writing it's version 7.1.14.

```
wget http://dev.mysql.com/get/Downloads/MySQL-Cluster-7.2/mysql-cluster-gpl-7.2.12-linux2.6-i686.tar.gz/from/http://cdn.mysql.com/
```

If there's already mysql user and group it's safe to ignore errors from adduser and groupadd

```
adduser mysql
groupadd mysql
```

```
cd /usr/local

tar zxvf mysql-cluster-gpl-7.2.12-linux2.6-i686.tar.gz

ln -s mysql-cluster-gpl-7.2.12-linux2.6-i686 mysql-cluster

cd mysql-cluster
```

```
chown -R mysql:mysql /usr/local/mysql-cluster/
mkdir -p /var/lib/mysql-mgmd-config-cache
mkdir -p /var/lib/mysql-mgmd-data
```

This is the minimum config.ini settings on management node. Refer MySQL Cluster Management Node Configuration Parameters and MySQL Cluster Data Node Configuration Parameters.

Important:
Setting NoOfReplicas to 1 means that there is only a single copy of all Cluster data; in this case, the loss of a single data node causes the cluster to fail because there are no additional copies of the data stored by that node.The maximum possible value is 4; currently, only the values 1 and 2 are actually supported.

Configure **"/var/lib/mysql-mgmd-data/config.ini**" file as below :

```
[ndbd default]
NoOfReplicas = 2
DataDir= /var/lib/mysql-ndb-data
DataMemory = 64M
IndexMemory = 128M

[ndb_mgmd]
# Management process options:
DataDir = /var/lib/mysql-mgmd-data
PortNumber = 2205
HostName = 192.168.2.53

[ndbd]
# Options for data node ndb1
hostname = 192.168.2.11

[ndbd]
# Options for data node ndb2
hostname = 192.168.2.60

[mysqld]
# SQL node options for sql1
hostname = 192.168.2.11

[mysqld]
# SQL node options for sql2
hostname = 192.168.2.60
```

Start Management Node.

```
cd /usr/local/mysql-cluster/bin

./ndb_mgmd --initial --configdir=/var/lib/mysql-mgmd-config-cache --config-file=/var/lib/mysql-mgmd-data/config.ini
```

That's it! you're done on management node. Grab a coffee if you can, the next steps will just be more on hopping to sql node and data node servers.

For each data node server, logon to each and perform the commands below. Assuming this is on ssh console connection, first logon to ndb1.

```
cd /opt

wget http://dev.mysql.com/get/Downloads/MySQL-Cluster-7.2/mysql-cluster-gpl-7.2.12-linux2.6-i686.tar.gz/from/http://cdn.mysql.com/

adduser mysql
groupadd mysql

cd /usr/local

tar zxvf /opt/mysql-cluster-gpl-7.2.12-linux2.6-i686.tar.gz

ln -s mysql-cluster-gpl-7.2.12-linux2.6-i686 mysql-cluster

cd mysql-cluster

chown -R mysql:mysql /usr/local/mysql-cluster/

scripts/mysql_install_db --user=mysql --datadir=/var/lib/mysql-ndb-data

bin/ndbd --ndb-connectstring=192.168.2.53:2205
```

Repeat the commands above on ndb2 data node servers.

You can now login to mgr1 node via ssh console and check the status of the nodes...

```
cd /usr/local/mysql-cluster

bin/ndb_mgm --ndb-connectstring=192.168.2.53:2205

show
```

For each sql node servers sql1, sql2 run the commands below

```
cd /opt

wget http://dev.mysql.com/get/Downloads/MySQL-Cluster-7.2/mysql-cluster-gpl-7.2.12-linux2.6-i686.tar.gz/from/http://cdn.mysql.com/

adduser mysql

groupadd mysql

cd /usr/local

tar zxvf mysql-cluster-gpl-7.2.12-linux2.6-i686.tar.gz

ln -s mysql-cluster-gpl-7.2.12-linux2.6-i686 mysql-cluster

cd mysql-cluster

chown -R mysql:mysql /usr/local/mysql-cluster/

scripts/mysql_install_db --user=mysql --datadir=/var/lib/mysql-node-data \
--basedir=/usr/local/mysql-cluster
```

Configure “/etc/mysqld-cluster.cf” file as below:

```
[mysqld]
ndbcluster
datadir=/var/lib/mysql-node-data
basedir=/usr/local/mysql-cluster
port=5000
ndb-connectstring=192.168.2.53:2205
```

```
bin/mysqld_safe --defaults-file=/etc/mysqld-cluster.cf --user=mysql &
```

Now on mgr1 node check the status of the nodes

ndb_mgm output from show command with all data nodes and sql nodes up and running.

```
/usr/local/mysql-cluster/bin/ndb_mgm --ndb-connectstring=192.168.2.53:2205

-- NDB Cluster -- Management Client --
ndb_mgm> show
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=2    @192.168.2.11  (mysql-5.5.25 ndb-7.3.0, Nodegroup: 0)
id=3    @192.168.2.60  (mysql-5.5.25 ndb-7.3.0, Nodegroup: 0, Master)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @192.168.2.53  (mysql-5.5.25 ndb-7.3.0)

[mysqld(API)]   2 node(s)
id=4  @192.168.2.11    (mysql-5.5.25  ndb-7.3.0)
id=5   @192.168.2.60   (mysql-5.5.25  ndb-7.3.0)
id=5   @192.168.2.60  (mysql-5.5.25 ndb-7.3.0)
```

To test the setup :-

Login to any of the sql nodes, in this example sql1(192.168.2.11) is used and execute the commands below assuming you're on ssh console connection.

```
CREATE DATABASE testdb;
USE testdb;
CREATE TABLE tblCustomer (ID INT) ENGINE=NDBCLUSTER;
INSERT INTO tblCustomer VALUES (1);
SELECT * FROM tblCustomer;
```

pay attention to the create table statement, we must specify ENGINE=NDBCLUSTER for all tables that we want to clustered. As stated earlier, MySQL cluster only saupports NDB engine, so if you use any other engine, table simply wont get clustered.

The result of the SELECT statement would be:

```
mysql> SELECT * FROM tblCustomer;
+------+
| ID   |
+------+
| 1    |
+------+
```

Since clustering in MySQL is at the "table level" not at the database level, so we would have to create the database sperately on sql2 (192.168.2.60) as well, but afterwards tblCustomer would be replicated with all its data (since the engine is NDBCLUSTER):

On sql2:

```
mysql -u root -p

CREATE DATABASE testdb;

USE testdb;

SELECT * FROM tblCustomer;
```

Now, if we insert a row of data on sql1, it should be replicated back to sql2:

```
mysql> INSERT INTO tblCustomer VALUES (2);
```

If we run a SELECT query on sql2, here is what we should see:

```
mysql> SELECT * FROM testtable;

+------+
| ID   |
+------+
| 1    |
| 2    |
+------+
```

Test Node shutdown:

Now run the following on 192.168.2.60 to test what happens if a node goes offline:

```
killall ndbd
```

and run this command to make sure that all ndbd processes have terminated:

```
ps aux | awk '!/awk/ && /ndbd/'
```

If you still see any processes, run this again:

```
killall ndbd
```

Now, let's go the management server (192.168.2.53) and run the following to check the cluster status:

```
/usr/local/mysql-cluster/bin/ndb_mgm --ndb-connectstring=192.168.2.53:2205
```

On the ndb_mgm console. run:

```
show;
```

it should be bring you an output simlar to the following:

```
ndb_mgm> show
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=2    @192.168.2.11  (mysql-5.5.25 ndb-7.3.0, Nodegroup: 0, Master)
id=3 (not connected, accepting connect from 192.168.2.60)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @192.168.2.53  (mysql-5.5.25 ndb-7.3.0)

[mysqld(API)]   2 node(s)
id=4    @192.168.2.11  (mysql-5.5.25 ndb-7.3.0)
id=5    @192.168.2.60  (mysql-5.5.25 ndb-7.3.0)
```

You see, 192.168.2.60 is not connected anymore.

Type quit; to leave the ndb_mgm management console. Now, let's check on 192.168.2.11, if our database is still up and we can make connections to it:

```
mysql -u root

USE testdb;

SELECT * FROM tblCustomer;

quit;
```

It should bring up the following result set:

```
mysql> SELECT * FROM tblCustomer;

+------+
| ID   |
+------+
| 1    |
| 2    |
+------+
```

Now, let's start MySQL on 192.168.2.60 again by issuing the following command:

```
/usr/local/mysql-cluster/bin/ndbd  --ndb-connectstring=192.168.2.53:2205
```

How to Restart MySQL Cluster:

In managing a produciton MySQL environment or any other transactional database environment, times come when we have to restart/shutdone our systems. So, let's see how would we shutdown our MySQL Cluster:

On Server1, open the management console:

```
ndb_mgm
```

then type:

```
shutdown;
```

it would bring up an output like this:

```
ndb_mgm> shutdown
Connected to Management Server at: 192.168.2.53:2205
Node 2: Cluster shutdown initiated
Node 3: Cluster shutdown initiated
Node 2: Node shutdown completed.
Node 3: Node shutdown completed.
3 NDB Cluster node(s) have shutdown.
Disconnecting to allow management server to shutdown.
```

This means that the cluster nodes 192.168.2.60 and 192.168.2.11 and also the Management node (192.168.2.53) have shut down.

To start the cluster management server again, run the following (on Server1, Management Server):

```
/usr/loca/mysql-cluster/bin/ndb_mgmd --initial --configdir=/var/lib/mysql-mgmd-config-cache --config-file=/var/lib/mysql-mgmd-data/config.ini
```

```
/usr/local/mysql-cluster/bin/ndb_mgm --ndb-connectstring=192.168.2.53:2205
```

```
cd /usr/local/mysql-cluster/bin
```

On 192.168.2.11

```
ndbd  --ndb-connectstring=192.168.2.53:2205
```

```
ndb_mgm> show
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=2    @192.168.2.11  (mysql-5.5.25 ndb-7.3.0, Nodegroup: 0, Master)
id=3    @192.168.2.60  (mysql-5.5.25 ndb-7.3.0, Nodegroup: 0)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @192.168.2.53  (mysql-5.5.25 ndb-7.3.0)

[mysqld(API)]   2 node(s)
id=4    @192.168.2.11  (mysql-5.5.25 ndb-7.3.0)
id=5    @192.168.2.60  (mysql-5.5.25 ndb-7.3.0)
```

phpMyAdmin

phpMyAdmin is a free and open source tool used to handle MySQL database functionalities via web browser. Creating, editing or dropping databases, fields or tables can be done in this tool with a click of a button (instead of typing long queries).

Installation Steps

Step 1 » Install/enable EPEL repository . You can find latest repository here (http://download.fedoraproject.org/pub/epel/)

```
rpm -Uvh http://www.mirrorservice.org/sites/dl.fedoraproject.org/pub/epel//5/i386/epel-release-5-4.noarch.rpm
```

Now you can install phpmyadmin and apache package

```
yum install httpd phpMyAdmin
```

Open "`/etc/httpd/conf.d/phpMyAdmin.conf`" file and find the lines “Deny from All ” and comment those lines and restart httpd service

Open "**`/etc/phpMyAdmin/config.inc.php`**" file and find the line that contain “blowfish_secret”, and modify like below.

```
$cfg['blowfish_secret'] = 'TypeAnything_for_Secure';
```

Now restart httpd service

Enter the URL **http://YOUR_SERVER_IP/phpMyAdmin/** on Firefox browser, we’ll can see the login web page.

ALTNIX Research & Development Team

- □ Mr Sadhiq Basheer Ahmed
- □ Mr Rahul Walunj
- □ Mr Rahul Patil
- □ Mr Venood khatuva
- □ Mr Javed Shaikh

MySQL Version 1.0

Author – Rahul Patil

ALTNIX RESEARCH AND DEVLOPEMENT TEAM