Replication

Index

1. What is Replication

1. Replication is a set of technologies for copying and distributing data and database objects from one database to another and then synchronizing between databases to maintain consistency. Using replication, you can distribute data to different locations and to remote or mobile users over local and wide area networks, dial-up connections, wireless connections, and the Internet.

2. Replication may be defined as a duplicate copy of similar data on the same or a different platform. The process of duplicating the data records in one database to one or more other databases in near-real time, is known as replication. The data can be presented in different formats.

Replication not only copies a database but also synchronizes a set of *replicas* so that changes made to one replica are reflected in all the others.

The beauty of replication is that it enables many users to work with their own local copy of a database but have the database updated as if they were working on a single, centralized database. For database applications where users are geographically widely distributed, replication is often the most efficient method of database access.

2. Types of Replication

Types of Replications in Oracle::

Oracle Server supports two different forms of replication: basic and advanced replication.

Basic replication is implemented using standard CREATE SNAPSHOT or CREATE MATERIALIZED VIEW statements. It can only replicate data (not procedures, indexes, etc), replication is always one-way, and snapshot copies are read only.

Advanced replication supports various configurations of updateble-snapshot, multi-master and update anywhere replication. It is more difficult to configure but allows data and other database objects like indexes and procedures to be replicated.

Types of Replication

Oracle's four basic types of replication. Starting from Oracle9 Read-only Snapshots are so called **Read-Only Materialized Views.**

Replication Type	Description
Read-only materialized views	A master table is copied to one or more databases. Changes in the master table are reflected in the snapshot tables whenever the snapshot refreshes. The snapshot site determines the frequency of the refreshes; data is pulled.
Updateable materialized views	Similar to read-only snapshots, except that the snapshot sites are able to modify the data and send their changes back to the master. The snapshot site determines the frequency of the refreshes and the frequency with which updates are sent back to the master.
Multi-master replication	A table is copied to one or more databases, and each database has the ability to insert, update, or delete records from it. Modifications are pushed to the other database at an interval that the DBA sets for each replication group. The highest theoretical frequency is once per second.

Replication Type	Description
Procedural replication	A call to a packaged procedure or function is replicated to one or more databases.

As you can see, these modes of replication are quite different, and each is suited for specific kinds of uses. A single environment can utilize all of these methods; they are not mutually exclusive.

Types of Replications in Mysql ::

Master – Slave replication ::

In most implementations of database replication, one database server maintains the master copy of the database and additional database servers maintain slave copies of the database.

Database writes are sent to the master database server and are then replicated by the slave database servers.

Database reads are divided among all of the database servers, which results in a large performance advantage due to load sharing.

In addition, database replication can also improve availability because the slave database servers can be configured to take over the master role if the master database server becomes unavailable.

Master – Master Replication ::

We need to replicate MySQL servers to achieve high-availability (HA). We need two masters that are synchronized with each other so that if one of them drops down, other could take over and no data is lost. Similarly when the first one goes up again, it will still be used as slave for the live one.

3. Uses of Replication

Replication can improve performance and increase availability of applications because alternate data access options becomes available. For example, users can access a local database rather than a remote server to minimize network traffic. Furthermore, the application can continue to function if parts of the distributed database are down as replicas of the data might still accessible.

Some of the most common uses of replication are described as follows:

Availability

Replication provides fast, local access to shared data because it balances activity over multiple sites. Some users can access one server while other users access different servers, thereby reducing the load at all servers. Also, users can access data from the replication site that has the lowest access cost, which is typically the site that is geographically closest to them.

Performance

Replication provides fast, local access to shared data because it balances activity over multiple sites. Some users can access one server while other users access different servers, thereby reducing the load at all servers. Also, users can access data from the replication site that has the lowest access cost, which is typically the site that is geographically closest to them.

Disconnected computing

A **materialized view** is a complete or partial copy (replica) of a target table from a single point in time. Materialized views enable users to work on a subset of a database while disconnected from the central database server. Later, when a connection is established, users can synchronize (refresh) materialized views on demand. When users refresh materialized views, they update the central database with all of their changes, and they receive any changes that may have happened while they were disconnected.

Network load reduction

Replication can be used to distribute data over multiple regional locations. Then, applications can access various regional servers instead of accessing one central server. This configuration can reduce network load dramatically.

Mass deployment

Replication can be used to distribute data over multiple regional locations. Then, applications can access various regional servers instead of accessing one central server. This configuration can reduce network load dramatically.

The target uses for replication in MySQL include:

- Scale-out solutions - spreading the load among multiple slaves to improve performance. In this environment, all writes and updates must take place on the master server. Reads, however, may take place on one or more slaves. This model can improve the performance of writes (since the master is dedicated to updates), while dramatically increasing read speed across an increasing number of slaves.

- Data security - because data is replicated to the slave, and the slave can pause the replication process, it is possible to run backup services on the slave without corrupting the corresponding master data.

- Analytics - live data can be created on the master, while the analysis of the information can take place on the slave without affecting the performance of the master.

- Long-distance data distribution - if a branch office would like to work with a copy of your main data, you can use replication to create a local copy of the data for their use without requiring permanent access to the master

4. Replication in WebNMS

The Web NMS has two BE servers :

- Primary BE
- Standby BE

In failover setup, the primary and standby servers should have access to the same or different(in case of replication) database. At any time, only one primary server and one standby server can be configured. In the database, details regarding the primary and secondary servers are maintained in a table named BEFailOver.

At a specified regular time interval, the primary server and secondary server updates LASTCOUNT using HEART_BEAT_INTERVAL .

When the primary server fails to update the LASTCOUNT, consecutive counts will be the same and the standby assumes that the primary had failed. Here, a parameter named RETRY_COUNT comes into play which enables us to specify the number of times the standby has to check the primary's LASTCOUNT (when the primary fails to update the LASTCOUNT) before assuming that the primary had failed.

The standby server of Web NMS offers warm standby support. Any operation or request in the Network during the intervening period (i.e., the time period between the failure of primary and the subsequent complete take over by standby) will be lost.

In Web NMS, two-way database replication is used, where it is like chained replication servers. That is, the Master and the Slave act as each other's Master and Slave.

Hot Standby Failover ::

Hot Standby Failover Support has been provided for Fault Management service. During failover period, the traps/notifications that are sent from the agent can be received and processed by the standby server's Fault Management service. Hence there is no loss of notifications during failover period.

5. Procedure for master-master replication

We will call system 1 as master1 and slave2 and system2 as master2 and slave 1.

Here is a basic step by step tutorial, that will cover the mysql master and master replication .

Notions: *we will call system 1 as master1 and slave2 and system2 as master2 and slave 1.*

Start mysql using command from <mysql home >

1. *./bin/mysqld_safe -- datadir=/<mysql home>/data/ --socket=/tmp/mysql.sock*

Step 1:

Install mysql on master 1 and slave 1. configure network services on both system, like

Master 1/Slave 2 ip: 192.168.111.46

Master 2/Slave 1 ip : 192.168.117.78

Step 2:

On Master2/Slave 1, edit my.cnf and master entries into it:

[mysqld]

datadir=<mysql home>/data

socket=/tmp/mysql.sock

server-id=2

report-host=192.168.111.46

master-host = 192.168.111.46

master-user = root

#master-password = slave

master-port = 3306

log-bin #information for becoming master added

binlog-do-db=webnms1

Step 3:

Create a replication slave account on master2 for master1:

mysql> GRANT REPLICATION SLAVE ON *.* TO ' <master-user> '@'%' ;

(Grant commands are persistence for long time.)

Step 4:

Edit my.cnf on master1 for information of its master.

[mysqld]

datadir=<mysql home>/data

socket=/tmp/mysql.sock

log-bin

binlog-do-db=webnms1 # input the database which should be replicated

server-id=1

#information for becoming slave.

report-host=192.168.117.78

master-host=192.168.117.78

master-user=root

#master-password = slave2

master-port=3306

Step 5:

Restart both mysql master1 and master2.

I*ssue the slave commands after logging in as root.(both master1, master2)*

1. mysql> stop slave;
*2. mysql > grant all privileges on *.* to 'root'@'%';*
3. mysql > flush privileges;
4. mysql> CHANGE MASTER TO MASTER_HOST=' master host',MASTER_USER=' master user',MASTER_PASSWORD=' ',MASTER_LOG_FILE='master log file',MASTER_LOG_POS=position ;(Look at output of mysql>show master status; and get these details)
5. start slave;

Slave_IO_Running and Slave_SQL_Running: must be to YES. Now you can create tables in the database and you will see changes in slave.

During replication the MySQL server creates a number of files that are used to hold the relayed binary log from the master, and record information about the current status and location within the relayed log. There are three file types used in the process:

- The *relay log* consists of the events read from the binary log of the master. Events in this binary log are executed on the slave as part of the replication thread.

- The *master.info* file contains the status and current configuration

information for the slave's connectivity to the master. The file holds information on the master hostname, login credentials, and the current position within the master's binary log.

- The *relay.info* file holds the status information about the execution point within the slave's relay log files.

The relationship between the three files and the replication process is as follows. The master.info file retains the point within the master binary log that has been read from the master. These read events are written to the relay log. The relay.info file records the position within the relay log of the statements that have been executed.

// sample my.cnf

#The options in this file will be used based on the script you call to start MySQL. SO if you start MySQL with safe_mysqld, (which is configured in startMySQL.sh), the options has to be configured under ***[mysqld] (Internally mysqld is called from safe_mysqld)***

[mysqld]
datadir=<mysql>/data
socket=/tmp/mysql.sock

[mysql.server]
user=mysql
basedir=/var/lib

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

If you start the mysql with the command mysqld, this will take the first set of code in the my.cnf and if you start the mysql server by using mysqld_safe command, then the 3rd set of code will take effect.

6. Replication Startup Options

This section describes the options that you can use on slave replication servers. You can specify these options either on the command line or in an option file.

On the master and each slave, you must use the `server-id` option to establish a unique replication ID. For each server, you should pick a unique positive integer in the range from 1 to 2^{32} – 1, and each ID must be different from every other ID. Example: server-id=3

Some slave server replication options are handled in a special way, in the sense that each is ignored if a `master.info` file exists when the slave starts and contains a value for the option. The following options are handled this way:

- `--master-host`
- `--master-user`
- `--master-password`
- `--master-port`
- `--master-connect-retry`
- `--master-ssl`
- `--master-ssl-ca`
- `--master-ssl-capath`
- `--master-ssl-cert`
- `--master-ssl-cipher`
- `--master-ssl-key`

7. Verification of replication setup

Checking that replication is still working ::

mysql> show slave status\G; *(master1/slave2)*

```
*************************** 1. row ***************************
             Master_Host: 192.168.117.78
             Master_User: root
             Master_Port: 3306
           Connect_retry: 60
         Master_Log_File: nmstest-lap-bin.026
     Read_Master_Log_Pos: 79
          Relay_Log_File: govardhini-relay-bin.001
           Relay_Log_Pos: 4
   Relay_Master_Log_File: nmstest-lap-bin.026
        Slave_IO_Running: Yes
       Slave_SQL_Running: Yes
         Replicate_do_db:
     Replicate_ignore_db:
              Last_errno: 0
              Last_error:
            Skip_counter: 0
     Exec_master_log_pos: 270
         Relay_log_space: 49
1 row in set (0.00 sec)
```

Slave_IO_Running and Slave_SQL_Running: must be to YES. Check the highlighted rows, make sure its running.

8. FAQ

1. What is replication and why should one use it?

Replication is the process of creating and maintaining replica versions of database objects (e.g. tables) in a distributed database system.

Replication can improve performance and increase availability of applications because alternate data access options becomes available. For example, users can access a local database rather than a remote server to minimize network traffic. Furthermore, the application can continue to function if parts of the distributed database are down as replicas of the data might still accessible.

2. About master.info

Once a slave is replicating, you will find a file called `master.info` *in the same directory as your error log. The* `master.info` *file is used by the slave to keep track of how much of the master's binary log is has processed.* ***Do not*** *remove or edit the file, unless you really know what you are doing. Even in that case, it is preferred that you use* `CHANGE MASTER TO` *command.*

3. Problem with Master_Log_File: <master log file>

check with

mysql> ***CHANGE MASTER TO***

```
->     MASTER_HOST='master_host_name',
->     MASTER_USER='master_user_name',
->     MASTER_PASSWORD='master_pass',
->     MASTER_LOG_FILE='recorded_log_file_name',
->     MASTER_LOG_POS=recorded_log_position;
```

4. Problem with master-port

If the master is listening on a non-standard port, you will also need to specify this with `master-port` *parameter in* `my.cnf`.

5. Problem with permissions

*Make sure that you logged as root execute ./bin/mysql -u root -h localhost and grant permission for master by executing mysql> grant all privileges on *.* to 'root'@'%'; and mysql>flush privileges;*

9. Links/Reference

Links::

1. *MySQL Master Master Repliction Tutorial, This totorial describes how to set up MySQL master-master replication.*

http://www.howtoforge.com/mysql_master_master_replication

2. mysqld Command-line Options

http://www.mcs.vuw.ac.nz/technical/software/MySQL/manual_MySQL_Database_Administration.html

3. Replication Startup Options and Variables

http://dev.mysql.com/doc/refman/5.0/en/replication-options.html