Basics of MySQL Replication

MySQL replication operates on the basis of a master-slave relationship. The master server logs all changes to its database into binary log files. The slave server(s) connect to the master, fetch these binary log files, and apply the changes to replicate the data. This process is asynchronous, meaning the master does not wait for slaves to apply changes.

Contents :

Key concepts in MySQL replication include:

- **Master**: The MySQL server that generates binary log files.
- **Slave**: The MySQL server(s) that replicate data from the master.
- **Binary Log**: Log files on the master server that record changes to databases.

Setting Up Replication in MySQL

Here are the basic steps to set up replication in MySQL:

1. **Configure the Master Server**
   First, configure the master server to enable binary logging and create a replication user with appropriate privileges.

A) **Enable Binary Logging**: Edit your MySQL configuration file (typically my.cnf or my.ini) to enable binary logging. Add or modify the following lines under the [mysqld] section:

```
[mysqld]
Server_id = 1
log_bin = /var/log/mysql/mysql-bin.log
```

- Server_id: Unique identifier for the master server. Each server in the replication topology must have a unique Server_id.
- log_bin: Path and filename for the binary log file.

1. **Restart MySQL**: Restart the MySQL service for the changes to take effect.
2. **Create a Replication User**: Create a MySQL user that the slave server(s) will use to connect to the master and replicate data.

```
/--- mysql ---/
Mysql> CREATE USER 'repl_user'@'slave_ip_address' IDENTIFIED BY 'your_password';
Mysql> GRANT REPLICATION SLAVE ON *.* TO 'repl_user'@'slave_ip_address';
Mysql> FLUSH PRIVILEGES;
```

Replace 'slave_ip_address' and 'your_password' with the actual IP address of the slave and a strong password.

3. **Lock the Master Databases**: Before taking a snapshot of the data to initialize the slave, lock the databases to ensure data consistency.

/-- mysql --/
Mysql> FLUSH TABLES WITH READ LOCK;
Mysql> SHOW MASTER STATUS;

Note down the File and Position from the output of SHOW MASTER STATUS;. You will need these values later when configuring the slave.

2. Configure the Slave Server

Next, configure the slave server to connect to the master and replicate data.

A) **Edit MySQL Configuration**: Edit your MySQL configuration file (my.cnf or my.ini) on the slave server to configure replication settings. Add or modify the following lines under the [mysqld] section:

/-- mysql --/
[mysqld]
Server_id = 2

A) Server_id: Unique identifier for the slave server. Must be different from the master's Server_id.

1. **Restart MySQL**: Restart the MySQL service on the slave server.
2. **Set Up Replication**:

/-- mysql --/

```
Mysql> CHANGE MASTER TO
       MASTER_HOST = 'master_ip_address',
       MASTER_USER = 'repl_user',
       MASTER_PASSWORD = 'your_password',
       MASTER_LOG_FILE = 'mysql-bin.xxxxxx',    -- File from SHOW MASTER STATUS on master
       MASTER_LOG_POS = xxxxxxx;                -- Position from SHOW MASTER STATUS on master
```

B) Replace 'master_ip_address', 'repl_user', 'your_password', 'mysql-bin.xxxxxx', and xxxxxxx with the actual values obtained from the master server.
3. **Start Replication**:

/-- mysql --/
Mysql> START SLAVE;

4. **Verify Replication Status**:

/-- mysql --/
Mysql> SHOW SLAVE STATUS\G;

C) Check the Slave_IO_State and Slave_IO_Running fields to ensure replication is working correctly.

2. Testing Replication
To test your replication setup:

D) **Insert Data**: Insert some data into the master server's database.

/-- mysql --/

Mysql> USE your_database;
Mysql> INSERT INTO your_table (column1, column2) VALUES ('value1', 'value2');

E) **Verify Data on Slave**: Check if the data has been replicated to the slave server.

/-- mysql --/
Mysql> USE your_database;
Mysql> SELECT * FROM your_table;

Additional Considerations

- **Monitoring**: Regularly monitor the replication status using SHOW SLAVE STATUS\G;.
- **Backup**: Back up your databases regularly, and ensure backups include both master and slave.
- **Security**: Secure your replication by using SSL and firewall rules to restrict access to MySQL ports.

Key Points of Database-Level Replication

· **Granularity**: MySQL replication at the database level replicates all tables within the specified databases. You cannot choose to replicate only certain tables from a database.

· **Efficiency**: Replication at the database level is efficient for applications that require a complete copy of the database.

· **Simplicity**: Setting up and managing replication at the database level is straightforward and generally easier compared to table-level replication.

Table-Level Replication

If you need to replicate only specific tables or a subset of data from a database, MySQL does not natively support table-level replication through its built-in replication mechanisms. However, you can achieve a similar result using techniques such as:

· **Filtered Replication**: Using replication filters (replicate-do-table, replicate-ignore-table) to include or exclude specific tables from replication. This is still considered database-level replication but allows you to filter out tables you don't want to replicate.

· **Custom Solutions**: Implementing custom solutions using triggers, stored procedures, or application-level logic to replicate specific tables or data subsets.

Option 1: Filtered Replication on master

-- Include specific table

replicate-do-table = your_database.your_table

-- Optionally, you can ignore other tables if needed

replicate-ignore-table = your_database.another_table

START SLAVE;

SHOW SLAVE STATUS\G;

Summary of Replication Filtering Options

- replicate-do-db=database_name: Replicate only the specified database.
- replicate-ignore-db=database_name: Ignore the specified database.
- replicate-do-table=database_name.table_name: Replicate only the specified table.
- replicate-ignore-table=database_name.table_name: Ignore the specified table.
- replicate-wild-do-table=database_name.table_pattern: Replicate tables matching the pattern in the specified database.
- replicate-wild-ignore-table=database_name.table_pattern: Ignore tables matching the pattern in the specified database.

Steps to Remove a Table from Replication

Stop the Slave Threads:

First, stop the replication threads on the slave server to ensure no changes are being applied while you update the configuration.

Mysql> STOP SLAVE;

Update Replication Filters on the Slave:

Modify the replication filters on the slave to ignore the specific table. You can do this by editing the my.cnf (or my.ini on Windows) file.

Open the MySQL configuration file on the slave server (usually located at /etc/mysql/my.cnf or /etc/mysql/mysql.conf.d/mysqld.cnf):

sudo nano /etc/mysql/my.cnf

Add or update the replicate-ignore-table option to exclude the table you want to remove from replication. For example, to ignore your_database.your_table:

[mysqld]

replicate-ignore-table = your_database.your_table

Restart the Slave Server:

Restart the MySQL service on the slave server to apply the configuration changes.

sudo systemctl restart mysql

Start the Slave Threads:

Start the replication threads again.

Mysql > START SLAVE;

Verify Replication Status:

Check the replication status to ensure it is running correctly and the table is no longer being replicated.

Mysql> SHOW SLAVE STATUS\G;

Example Configuration Change

[mysqld]

server-id = 2

log_bin = /var/log/mysql/mysql-bin.log

Replication filters to ignore specific tables

replicate-ignore-table = your_database.your_table

Steps to IGNORE DB replication on SLAVE SIDE

Master server

binlog-ignore-db = * # Add the following line to exclude all databases from the binary log by default:

Steps to Remove Replication

1. **Stop the Slave Threads**: Stop the replication threads on the slave server.

Mysql> **STOP SLAVE;**

2. **Reset the Slave Configuration**: Reset the slave configuration to remove the master information and other replication settings.

Mysql> **RESET SLAVE ALL;**

RESET SLAVE ALL will remove all replication-related configuration settings, including the master information. Be cautious with this command as it cannot be undone.

3. **Remove Replication User on the Master**: If the replication user created for the slave is no longer needed, you can remove it from the master server.

Mysql> DROP USER 'repl_user'@'slave_ip_address';

Mysql> FLUSH PRIVILEGES;

4. **Clean Up Replication Logs (Optional)**: If you want to remove the binary logs on the master server, you can purge them. However, be cautious with this step, as it will remove the logs permanently.

Mysql> PURGE BINARY LOGS TO 'mysql-bin.xxxxxx'; -- Replace with the appropriate log file

5. **Restart MySQL (Optional)**: If you made changes to the configuration files or want to ensure all settings are applied cleanly, you can restart the MySQL server on the slave.

sudo systemctl restart mysql

Steps to Set Up Replication for a Large Database

Ensure Sufficient Resources:

- Ensure the slave server has adequate resources (CPU, memory, storage) to handle the replication load.
- Verify network bandwidth between the master and slave servers can handle large data transfers.

Prepare the Master and Slave Servers:

- Ensure both the master and slave servers are running the same version of MySQL.
- Configure the master server with the necessary binary logging options.

On the master (`my.cnf`):

[mysqld]

server-id = 1

log_bin = /var/log/mysql/mysql-bin.log

On the slave (my.cnf):

[mysqld]

server-id = 2

Create a Replication User:

Lock the Master Tables (Optional but Recommended):

Mysql> FLUSH TABLES WITH READ LOCK;

Mysql> SHOW MASTER STATUS;

Note the File and Position values from the SHOW MASTER STATUS output. You will need these for configuring the slave.

Dump the Database: Use mysqldump to create a consistent snapshot of the master database. Compress the dump to save space and speed up the transfer.

mysqldump --single-transaction --master-data=2 --all-databases | gzip > dump.sql.gz

Transfer the Dump to the Slave: Transfer the dump file to the slave server using a tool like scp or rsync.

scp dump.sql.gz user@slave_ip:/path/to/dump/

Import the Dump on the Slave: Import the dump file on the slave server.

gunzip < dump.sql.gz | mysql

Configure the Slave: Configure the slave server with the master server details and the binary log coordinates noted earlier.

Mysql> CHANGE MASTER TO

MASTER_HOST = 'master_ip',

MASTER_USER = 'repl_user',

MASTER_PASSWORD = 'password',

MASTER_LOG_FILE = 'mysql-bin.000001', -- Replace with the actual log file name

MASTER_LOG_POS = 123456; -- Replace with the actual log position

Start the Slave: Start the replication process on the slave.

Mysql> START SLAVE;

Verify Replication Status:

Mysql> SHOW SLAVE STATUS\G;

Created By

Ajit Yadav (DBA)