Database Management Systems Training

June 12 - 16, 2023

One Central Hotel
Cebu City, Philippines

MySQL Replication

MySQL Replication

- MySQL replication is a feature that enables the replication of data from one MySQL database server, known as the master, to one or more MySQL database servers, known as the slaves.

- Replication allows for data synchronization and redundancy, improving performance, scalability, and fault tolerance in MySQL environments.

Some Key Points

- Master-Slave Architecture
- Replication Process
- Replication Modes
- Replication Configuration
- Replication Filters
- Replication Lag and Monitoring
- Failover and High Availability

Master-Slave Architecture

- In MySQL replication, there is a master database server that receives write operations (updates, inserts, deletes) and one or more slave database servers that replicate the changes from the master.

Replication Process

- The master records all data modifications in its binary log, which is a sequence of events that represent the changes to the database.

- The slave(s) connect to the master, retrieve the binary log events, and apply them to their own databases to mirror the changes.

Replication Modes

- MySQL supports different replication modes, including statement-based replication (SBR), row-based replication (RBR), and mixed-based replication (MBR), which combine elements of SBR and RBR.

- These modes determine how changes are replicated from the master to the slave(s).

Replication Configuration

- To set up replication, you need to configure the master and slave(s).
- The master server needs to enable binary logging, and the slave(s) need to specify the master server's details (host, port, credentials) to establish the replication connection.

Replication Filters

- MySQL allows you to specify replication filters to control which databases or tables should be replicated.
- This provides flexibility in choosing the data to be replicated based on specific requirements.

Replication Lag and Monitoring

- Replication can introduce some delay between the master and the slaves due to network latency or processing time.
- Monitoring tools and techniques can be used to measure replication lag and ensure it is within acceptable limits.

Failover and High Availability

- MySQL replication can be utilized for achieving high availability by configuring the slaves for failover.
- In case of a master failure, one of the slaves can be promoted as the new master to continue serving database operations.

Master-Slave Configuration

Step-1 : Prepare the Master Server

1. Open the MySQL configuration file (**my.cnf** or **my.ini**) on the master server.
2. Locate the variable **server-id** and set the value to 1
3. Add the following lines after the **server-id** setting:

```
log-bin=mysql-bin
binlog_format=ROW
binlog_row_image=full
```

Step-2 : Restart the Master Server

1. Restart the MySQL service on the master server to apply the configuration changes.

Step-3 : Create a Replication User on the Master Server

1. Connect to the MySQL server as a privileged user (e.g., root).

2. Create a new user for replication with appropriate privileges.
   Run the following commands:

```
CREATE USER 'replication_user'@'%' IDENTIFIED BY '1234'
;
GRANT REPLICATION SLAVE ON *.* TO 'replication_user'@'%'
;
```

Step-3 : Create a Replication User on the Master Server

Run the following commands:

```
CREATE USER 'replication_user'@'localhost' IDENTIFIED BY '1234'
;
GRANT REPLICATION SLAVE ON *.* TO 'replication_user'@'localhost'
;

FLUSH PRIVILEGES
;
```

Step-4 : Take a Snapshot or Perform a Backup

1. Take a snapshot of the master server's data or perform a backup to ensure the initial data consistency for replication.

Step-5 : Configure the Slave Server(s)

1. Open the MySQL configuration file (**my.cnf** or **my.ini**) on the master server.
2. Locate the variable **server-id** and set the value to 2
3. Add the following line after the **server-id** setting:

```
relay-log=mysql-relay-bin
```

Step-6 : Restart the Slave Server(s)

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

Step-7 : Start Replication on the Slave

1. Connect to the MySQL server on the slave.
2. Execute the following command to start the replication process:

```
CHANGE MASTER TO
    MASTER_PORT=13306,
    MASTER_HOST='localhost',
    MASTER_USER='replication_user',
    MASTER_PASSWORD='1234',
    MASTER_LOG_FILE='mysql-bin.000001',
    MASTER_LOG_POS=4
;
```

Step-8 : Start Replication on the Slave

1. Execute the following command on the slave to start replication:

```
START SLAVE
;
```

Step-9 : Verify Replication Status

1. Execute the following command on the slave to start replication:

```
SHOW SLAVE STATUS
;
```

2. Check the **Slave_IO_Running_State** and **Slave_SQL_Running** fields.

3. Ensure that both show Waiting for master to send event and Yes, respectively, to confirm successful replication.

Congratulations!

You have successfully set up MySQL replication using version 5.7

Changes made on the master server should now be replicated to the slave server(s).

Remember to monitor the replication process and ensure that it remains healthy.