Reading: Perform Point-in-Time Backup and Restore

Estimated Time: 10 minutes

Objectives:

After completing this reading, you will be able to:

- Understand the significance of Point-in-Time Recovery (PITR) in database management.
- Gain an overview of key Linux commands used for conducting Point-in-Time Backup and Restore of MySQL Database.

Significance of Point-in-Time Recovery (PITR) in Database Management:

Point-in-Time Recovery (PITR) allows you to restore a database to a specific moment, often before an error or data corruption occurred. This capability is critical for minimizing data loss and maintaining data integrity, especially in environments where continuous availability and data accuracy are paramount.

PITR involves a combination of full backups and incremental backups (or binary logs in MySQL), enabling administrators to revert the database to any desired point in time. This method provides flexibility and precision in recovery operations, ensuring that only the necessary data changes are applied during the restoration process.

Key Techniques for Point-in-Time Backup and Restore:

This section explores the essential techniques and methods used for performing PITR in MySQL database systems.

Full Backups and Incremental Backups:

PITR relies on both full backups and incremental backups to provide a comprehensive recovery solution. These two types of backups work together to ensure that you can restore your database to any specific point in time.

Full Backups
A full backup captures the entire database at a specific point in time. It includes all the data, schema, and configurations necessary to restore the database to that state. Full backups serve as the foundation for PITR and are typically performed periodically (e.g., daily or weekly) to ensure that there is a recent baseline from which to start the recovery process.

Incremental Backups
Incremental backups capture only the changes made to the database since the last backup (full or incremental). This approach reduces the amount of data that needs to be backed up, saving time and storage space. In MySQL, binary logs are crucial for incremental backups as they record all database changes.

Binary Logs

Binary logs in MySQL are essential for PITR. They record all database changes (e.g., INSERT, UPDATE, DELETE statements) in a binary format. These logs enable the replay of changes during recovery, allowing you to restore the database to any specific point in time by applying the changes recorded after the last full backup.

How Binary Logs Work

1. Enable Binary Logging: Binary logging must be enabled on the MySQL server. This can be done by adding the log_bin directive to the MySQL configuration file (my.cnf) and restarting the MySQL server.

2. Recording Changes: Once binary logging is enabled, MySQL starts recording all changes to the database in binary log files. Each log file is given a sequential number (e.g., mysql-bin.000001, mysql-bin.000002).

3. Managing Log Files: As the number of binary log files increases, it's important to manage them to prevent disk space issues. MySQL provides options to purge old binary logs that are no longer needed.

4. Point-in-Time Recovery: To perform a point-in-time recovery, you first restore the database from the most recent full backup. Then, you apply the changes recorded in the binary logs to bring the database to the desired state.

Practical Steps for Point-in-Time Backup and Restore:

Let us explore a scenario where you are tasked with taking a point in time backup of the MySQL server that houses a database named **world**. Within this database, you will encounter tables like **cities**, **countries** and **languages**, each holding data related to **cities**, **countries** and **languages** spoken worldwide.This exercise will build upon the world database created in the previous exercise: Backup and Restore Using MySQL.

Say you have a full logical backup of your whole database in your last mysqldump file as of yesterday evening. However, several changes may have been made (including data loss) since then. Using point-in-time backup and restore, you can get each and every change that occurred since then, so that even after your last logical backup you have a record of all new transactions. Point-in-time backup is the set of binary log files generated subsequent to a logical backup operation of a database. The binary log files contain events that describe database changes such as table creation operations or changes to table data. To restore a database to a point-in-time, you will be using binary log files containing changes of a database for a time interval along with the last logical backup of the database.

Performing Point-in-Time Backup:

Download the required SQL script file in the linux terminal by executing the command below:

```
wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0231EN-SkillsNetwork/datasets/World/world_mysql_script
```

Download the required SQL script file in the linux terminal by executing the command below:

```
wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0231EN-SkillsNetwork/datasets/World/world_mysql_update
```

Download the required SQL script file in the linux terminal by executing the command below:

```
wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0231EN-SkillsNetwork/datasets/World/world_mysql_update
```

Once executed, the above commands will fetch the SQL script file from the internet and save it to the current directory on your local machine.

Pre-requisite: To interact with MySQL databases on your local machine, you need to install and start the MySQL server. To install the MySQL server run the following command:

```
sudo apt-get install mysql-server
```

This command will prompt you to enter your password, and once you confirm, it will download and install the MySQL server package along with any necessary dependencies.

> Note: In case you encounter errors regarding unreachable packages, execute the command `sudo apt update` to fetch the most recent information concerning available packages and their versions from the repositories set up on your system. Once done you can then execute the command to install the mysql-server.

Start the MySQL service using the command:

```
sudo systemctl start mysql
```

This command is used to start the MySQL service on a system running systemd, which is a system and service manager for Linux.

Here's what each part of the command does:

- **sudo**: This prefix grants temporary administrative privileges to the command that follows it. It allows the user to execute the subsequent command with superuser (root) privileges.
- **systemctl**: This is a command-line utility used to manage **systemd**, which is a system and service manager for Linux operating systems. It allows users to start, stop, enable, disable, and manage services and other units.
- **start**: This subcommand of **systemctl** instructs systemd to start the specified service. In this case, **mysql** refers to the MySQL service.

Next, to access the MySQL CLI(Command Line Interface), you can use the command below in the terminal:

```
sudo mysql -u root -p
```

Here's what each part of the command does:

- **mysql**: This is the command used to launch the MySQL command-line interface.

- **-u root**: This option specifies the username to use when connecting to the MySQL server. In this case, it specifies that the root user should be used.
- **-p**: This option prompts the user to enter the password for the specified MySQL user (in this case, the root user). After entering the password, if it's correct, the user gains access to the MySQL CLI with administrative privileges.

Create a new database world using the command below in the terminal:

```
create database world;
```

To use the newly created world database of previous exercise, use the command below in the terminal:

```
use world;
```

Execute the world mysql script (world_mysql.sql) to complete the world database creation process using the command below in the terminal:

```
source world_mysql_script.sql;
```

List all the table names from the world database:

```
SHOW TABLES;
```

Retrieve all records from the city table where the countrycode is ‘CAN’:

```
SELECT * FROM city WHERE countrycode='CAN';
```

You will observe the returned result set is empty set. This means Canada related records are currently absent from the table. Run the update script (world_mysql_update_A.sql) to insert the records you were looking for.

```
source world_mysql_update_A.sql;
```

Redo the previous step to verify the insertion of Canada-related records:

```
SELECT * FROM city WHERE countrycode='CAN';
```

Once the operation is completed, we can quit from the MySQL command line interface using the command `\q` in the terminal.

```
\q
```

Next, create a full logical backup of the current state of your whole world database. Use the command below in the terminal (enter your MySQL service session password from the MySQL service session tab if necessary):

```
sudo mysqldump  --user=root --password --flush-logs --delete-master-logs --databases world > world_mysql_full_backup.sql
```

Note: The –flush-logs and –delete-master-logs parameters ensure that new binary log files are created after the full backup.

Next, to access the MySQL CLI(Command Line Interface), you can use the command below in the terminal:

```
sudo mysql -u root -p
```

To use the already created above world database, use the command below in the terminal:

```
use world;
```

List all the table names from the world database:

```
SHOW TABLES;
```

Retrieve all records from the city table where the countrycode is 'CAN':

```
SELECT * FROM city WHERE countrycode='CAN';
```

You will observe the returned result set is empty set. This means Canada related records are currently absent from the table.Run the update script (world_mysql_update_B.sql) to insert the records you were looking for.

```
source world_mysql_update_B.sql;
```

Redo the previous step to verify the insertion of Canada-related records:

```
SELECT * FROM city WHERE countrycode='CAN';
```

Once the operation is completed, we can quit from the MySQL command line interface using the command `\q` in the terminal.

```
\q
```

Simulate a Database Crash:

Now you will create a scenario where a database crash will be conducted intentionally which will result a significant loss of your world database files.To simulate a database crash and create a scenario for recovery, execute the following commands to stop the MySQL service and remove the database files:

Stop the MySQL service using the following command:

```
sudo systemctl stop mysql
```

Here's what each part of the command does:

- **systemctl**: This is a command-line utility used to manage **systemd**, which is a system and service manager for Linux operating systems. It allows users to start, stop, enable, disable, and manage services and other units.
- **stop**: This subcommand of **systemctl** instructs systemd to stop the specified service. In this case, **mysql** refers to the MySQL service.

Remove the world database files using following command:

```
sudo rm -rf /var/lib/mysql/world
```

The `rm -rf` command removes the world directory and all of its contents recursively and forcefully. This includes all the data files, subdirectories, and any other files related to the world database within the MySQL data directory.

Try to retrieve records from any table of the database:

```
sudo mysql --user=root --password --execute="SELECT * FROM world.city;"
```

You will face errors since a significant loss of your **world** database files happened. Now you have to restore the world database along with the updates you made earlier in this exercise running the update script (world_mysql_update_B.sql).

Display the binary logs using the command below in the terminal:

```
sudo mysql --user=root --password --execute="SHOW BINARY LOGS;"
```

Write the contents of all binary log files listed above to a single file using the command below in the terminal:

```
sudo mysqlbinlog /var/lib/mysql/binlog.000003 /var/lib/mysql/binlog.000004 > logfile.sql
```

Here's what each part of the command does:

- Read Binary Log Files: The `mysqlbinlog` utility reads the specified binary log files (binlog.000003 and binlog.000004).
- Convert to SQL Statements: `mysqlbinlog` converts the binary format of the log files into readable SQL statements. These statements represent the database changes recorded in the logs.
- Redirect Output: The SQL statements are redirected and saved into a file named `logfile.sql`.

Performing Point-in-Time Recovery:

You are ready to perform point-in-time restore. First restore the full logical backup of your whole world database you created earlier in this exercise using the command below in the terminal (enter your MySQL service session password from the MySQL service session tab if necessary):

```
sudo mysql --user=root --password < world_mysql_full_backup.sql
```

To verify if you have the updates from the update script (world_mysql_update_B.sql), retrieve all the Canada (countrycode='CAN') related records from the **city** table using the command below in the terminal (enter your MySQL service session password from the MySQL service session tab if necessary):

```
sudo mysql --user=root --password --execute="SELECT * FROM world.city WHERE countrycode='CAN';"
```

Now run the logfile you created in step-14 using the command below in the terminal:

```
sudo mysql --user=root --password < logfile.sql
```

Redo previous step to verify if you have the updates from the update script (world_mysql_update_B.sql).

```
sudo mysql --user=root --password --execute="SELECT * FROM world.city WHERE countrycode='CAN';"
```

Finally through the point-in-time recovery, you have the **world** database in the same state before you conducted the intentional crash scenario.

Conclusion:

This reading highlights the critical role of Point-in-Time Recovery in database management, providing detailed steps and best practices for performing PITR in MySQL. By mastering these techniques, you can ensure robust data protection and minimize downtime in your database environment.

Author(s)

Pratiksha Verma

Other Contributors

Malika Singla, Lakshmi Holla