Binary Logging

==========

Binary Logging

==========

What is Binary Logging?

Binary logging in MySQL is a crucial feature for data replication, backup, and recovery. It logs all changes made to the database, such as updates, deletions, and insertions, in a binary format.

Purpose of Binary Logging

- **Replication**: Binary logs are used to replicate data from a primary server to one or more replica servers. Changes logged on the primary server are applied to the replicas to keep them synchronized.
- **Point-in-Time Recovery (PITR)**: In case of data corruption or accidental data loss, binary logs allow you to restore a backup and replay the transactions recorded in the binary logs to recover the database to a specific point in time.
- **Audit Trails**: They can be used to audit and review changes made to the database over time.

How Binary Logging Works

- **Binary Log Files**: The changes are recorded in binary log files (with extensions like `.000001`, `.000002`, etc.) which are stored in the MySQL data directory.
- **Log Events**: Each event in the binary log represents a single database change such as a query or a transaction. These events are recorded in the order they were executed.
- **Binary Log Index File**: MySQL maintains an index file (typically `mysql-bin.index`) that lists all the binary log files and their sequence.

==========

Binary Logging Configuration

Enabling Binary Logging

To enable binary logging, you need to configure the MySQL server with the `log_bin` parameter in the `my.cnf` configuration file:

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

Binary Logging Options

Bin log options that can also be set in the my.cnf file:

expire_logs_days = 7	Defines how many days to retain binary log files before automatic deletion.
binlog_format = format format: ROW STATEMENT MIXED	Determines the format of the binary logs. The available formats are: ROW:Logs the actual changes at the row level. STATEMENT: Logs each SQL statement that modifies data. MIXED: Uses a combination of statement and row formats.
max_binlog_size = 100M	Sets the maximum size of a binary log file before a new file is created.

Binary Log Management

Viewing Binary Logs

Use the `SHOW BINARY LOGS` or `SHOW MASTER LOGS` command to list the binary log files.

```
SHOW BINARY LOGS;
```

Examining Binary Log Content

Use the `mysqlbinlog` utility to examine the content of binary log files

```
mysqlbinlog /var/log/mysql/mysql-bin.000001
```

Purging Binary Logs

To manually delete old binary logs and free up space, use the `PURGE BINARY LOGS` statement.

```
PURGE BINARY LOGS TO 'mysql-bin.000010';
```

Or, you can purge logs older than a specific date:

```
PURGE BINARY LOGS BEFORE '2024-01-01 00:00:00';
```

==========

Data Recovery From Binary Logs

> Binary Logs are just transactional changes made to a database. They don't include a full copy of the database itself. In order to replay binary logs, you'll need a backup copy of the database to replay onto. See mysql backups.

1. Identify the Binary Logs to Use

Determine the range of binary logs that contain the transactions you need to replay. You can list all binary logs with:

```
SHOW BINARY LOGS;
```

2. Obtain a Backup

Obtain the newest backup/dump of the database prior to the time that we need to recover to.

3. Import the backup into MySQL

```
mysql -u root -p < /path/to/backup.sql
```

4. Apply Binary Logs

A. Identify the start and end points within the binary logs. You might need to apply all transactions or up to a specific point in time or transaction.

```
mysqlbinlog --start-datetime="2024-06-28 10:00:00" --stop-datetime="2024-06-29 14:00:00"
/path/to/mysql-bin.000001 /path/to/mysql-bin.000002 | mysql -u root -p
```

==========

Revision #4
Created 2024-06-29 16:58:03 UTC by Daniel
Updated 2024-06-29 17:41:07 UTC by Daniel