DAT151
Database and Unix System Management

Spring 2024

Assignment 8

Obligatory assignment. Deadline: Sunday 28.04.2024.

Your code should be easy to read:

- *Use indentation in your code.*
- *DDL for CREATE TABLE should have each column definition on its own line.*
- *Do not use screen dumps to show code. Code should be normal text, formatted as code.*

The report should include all necessary commands to complete the tasks, printout from the system, explanation of what is done, the result and explanation of the result.

Remember to include the names of the group members on the front page of the report. The report can be in English or Norwegian. The report should be handed in via it's learning.

The assignment should be accomplished in groups of two or three students. The assignment should be accomplished in groups of one to three students. Signing up for a group will be closed on Friday 26.04.

You must select a group "Lab8 N" when delivering to see all the comments that you will get on this assignment.

***Observe**: Task three will require a minimum of two lab computers, and a minimum group size of two is necessary. If you still prefer to work alone, then you must set up two computers at the lab running AlmaLinux 9.*

***Observe**: Before you modify an existing configuration file, save a copy of the original somewhere on the computer.*

Lecturers:
Bjarte Kileng <Bjarte.Kileng@hvl.no>, room E418
Haakon André Reme-Ness <Haakon.Andre.Reme-Ness@hvl.no>, Fabrikkgaten 5, R235-02

Last changed, 09.04.2024 (BK)

Task 1: Backup

In this task you will set up a backup procedure for your MariaDB server. The backup should be stored on a medium different from that of the disks storing the databases and indexes.

Important: Test you backup and recovery procedures first on databases with a small amount of data!

The backup should be done at regular intervals automatically e.g. as a cron job. The backup procedure must also handle the binary logs. Binary logs must regularly be removed from disk or the logs will fill the disk, but logs must not be deleted before they have been added to a backup.

Use the program *mysqldump* to make backups of the MariaDB databases . This program makes a logical backup. It can dump all SQLs necessary to recreate tables and fill the tables with data.

The utility *mysqldump* can make a backup of all databases simultaneously, but this is usually not a good solution. Recovery usually only involves a subset of databases and this is simpler if backup of each database is made individually.

If relations exist between objects from different databases, all the involved databases must be dumped simultaneously. We can remove this problem by restricting access from applications and users to only one database.

The backup system should do the following:

- Backup each database individually.
- Backup the binary logs.
- Issue the necessary FLUSH and PURGE of the binary logs.
- Copy the backup to a disk on a different computer.

Backup should be done by a script at scheduled intervals using the cron system. Check the manual pages for *crontab*.

When doing a backup, the databases and logs should be stored together. Tag the backups for easy retrieval. A good idea is to put the databases and logs in a tar file and name the file with the date and time of the backup. The file name can include a UNIX timestamp, but should also include the date and time in a more human readable format.

You need to determine the active binary log. This can be achieved with the "*--master-data=2*" option. The "*--master-data*" switch is best used together with the "*--single-transaction*" switch . You should also issue a "FLUSH LOGS" command to start a new binary log somewhere in the backup process.

What databases and tables should be included in the backup? The *information_schema* only contains views and should be skipped. Skip also the database *performance_schema*. The database *mysql* stores e.g. users and their grants. Your backup must include this information, but in a logical form that can be used for a later restore. The switch *--system* of *mysqldump* can be used. This switch let you dump different system tables as SQL that can be used to restore the data.

For all other databases, backup all database objects.

When doing backup of an individual database it might be useful to drop and recreate the database before restoring data from the backup. The program *mysqldump* will only put a "*CREATE DATABASE*" statement in the dump if used with options "*--all-databases*" or "*--databases*". When backing up individual databases, either use "*--databases*" or let the backup script add to the top of the dump:

```
CREATE DATABASE IF NOT EXISTS …;

USE …;
```

Task 2: Recovery

You will need some data in the database system before you test your backup system. Create at least three databases with tables.

- One database should store the tables of assignment 7.
- One database should store the tables of assignment 4, task 6.
- One database should store the tables of assignment 2, task 4.

Use the InnoDB engine for all tables to enforce referential integrity.

Perform the following steps on the databases:

1. Take backup of the databases. At this stage, table *Passing* from lab 7 has 4 154 196 rows.
2. Insert 1000 new rows into *Passing*. Table *Passing* now has 4 155 196 rows.
3. Drop table *Passing*.
4. Create an empty table *Passing* using the original DDL.
5. Insert 100 rows into *Passing*. Table *Passing* now has 100 rows.

Use the backups and the binary logs to restore the databases to current, but remove the effect of the transaction that deleted *Passing*. I.e. restore the databases as they would have been at current if you had not deleted and recreated *Passing* at stage three and four above.

Important: Always flush the binary log before loading data from backup!

The recovery must remove the effects of step three and four in the above list. The recovered *Passing* should have 4 154 196+1000+100 rows, or 4155296 rows.

Task 3: Replication

In this task you will set up database replication between a master and a slave. With replication, data from the database master will be copied to the slave. If the master crashes, the replication slave will have all the data and can replace the master.

The MariaDB manuals gives extensive documentation on installing and running a MariaDB master with replicating slaves, see the chapter on MariaDB Replication. You will use two computers when solving this task.

The replication in MariaDB is asynchronous meaning that a transaction is committed when the master successfully has committed the transaction. If the slave is not be working, or is slow, the slave can lag the master by several transactions.

Replication can be handled also by the Distributed Replicated Block Device (DRDB) Linux kernel module. DRBD can synchronously replicate storage, but we will not use DRDB in this assignment.

Set up a 2-way replication between two database servers, i.e. both servers should act as master and slave for the other. If one of the servers fail, the other server should have all the data.

Check the synchronisation status in both ends. Make sure both servers report that the replication as up and running.

Test the replication by manipulating data in the database you created in the first part of the assignment. Check that the changes replicate to the other computer and make sure to test this in both directions.