Adrian Mortensen

DAT151 – Oblig7

Backup and recovery

Contents

[Task 1: Backup 2](#_Toc33455553)

[Script: 2](#_Toc33455554)

[Crontab: 2](#_Toc33455555)

[Results: 2](#_Toc33455556)

[Task 2: Recovery 3](#_Toc33455557)

[Setup 3](#_Toc33455558)

[From dump 4](#_Toc33455559)

[Binary logs 4](#_Toc33455560)

[Results 4](#_Toc33455561)

[Task 3: Replication 5](#_Toc33455562)

# Task 1: Backup

## Script:

#!/bin/bash

#Faste variabler

BACKUPDIR**=**"/extra/backup/DB"

LOGDIR**=**"/var/lib/mysql"

NOW**=$( date '+%F\_%H:%M:%S' )**

TIMESTAMP**=$( date '+%s')**

**mkdir** **$BACKUPDIR/$TIMESTAMP**

#Backup av databaser

# Finner liste av databaser som kan kjøre vanlig prosedyre.

# Bruker brukeren backupAdmin som har rettigheter til å gjøre backup.

**set** **--** **$(mysql -u backupAdmin --skip-column-names -e "SHOW DATABASES WHERE \`Database\` NOT IN ('mysql','information\_schema','performance\_schema')")**

**for** db

**do**

mysqldump **--**user backupAdmin **--**master-data**=**2 **--**single-transaction **-**F **--**databases **$db** **>** **$BACKUPDIR/$TIMESTAMP/$db**'\_'**$NOW.**sql

**done**

# Egen backup av mysql databasen

mysqldump **--**user backupAdmin **--**single-transaction **-**F mysql **--**databases **>** **$BACKUPDIR/$TIMESTAMP/**'mysql\_'**$NOW.**sql

#Binær log backup:

#Flytter binærloggene

**mv** **$LOGDIR/\*-**bin.0**\*** **$BACKUPDIR/$TIMESTAMP/**

# lager tar.gz arkiv av backup for å spare plass.

**tar** **-**czvf **$BACKUPDIR/$TIMESTAMP**'\_Backup\_'**$NOW.**tar.gz **$BACKUPDIR/$TIMESTAMP**

# Sletter ukomprimert mappe

**rm** -r **$BACKUPDIR/$TIMESTAMP**

**exit** 0

## Crontab:

**$ sudo** EDITOR**=**nano crontab -e

no crontab for root **-** using an empty one

0 14 **\*** **\*** **\*** **/**home**/**admo**/**git**/**DAT151**/**Oblig7**/**Scripts**/**runBackup.sh

Adding a crontab for running the backup at 14:00 every day. (the time 14:00 is for testing purposes because it was added around 13:50.

Got this file:

Backup: 1582570801\_Backup\_2020-02-24\_14:00:01.tar.gz

## Results:

The result is archive files that contains the sql dumps of each database, a with timestamp both unix timestamp and human readable time for backup.

# Task 2: Recovery

## Setup

Following the task:

1. Take backup of database where table Passing has 100000 rows
   1. Done with the script
2. Insert 1000 rows into Passing
   1. Code for insertion:

MariaDB **[**Oblig5**]>** INSERT INTO Passing SELECT DISTINCT regno**,**now**(),**3 FROM Car LIMIT 1000**;**

Query OK**,** 1000 rows affected **(**0**.**019 sec**)**

Records**:** 1000 Duplicates**:** 0 Warnings**:** 0

1. Delete table Passing

* Code for deletion  
  MariaDB **[**Oblig5**]>** DROP TABLE Passing;

1. Create empty table Passing

* Code for table creation

**CREATE** **TABLE** **IF** **NOT** **EXISTS** Passing **(**

regno CHAR**(**7**)** **NOT** **NULL,**

timestamp TIMESTAMP **NOT** **NULL,**

tollstation SMALLINT UNSIGNED **NOT** **NULL,**

**PRIMARY** **KEY** **(** regno **,** timestamp **),**

**CONSTRAINT** fk\_Passing\_1

**FOREIGN** **KEY** **(** tollstation **)**

**REFERENCES** Tollstation **(** idTollstation **)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**,**

**CONSTRAINT** fk\_Passing\_Subscription1

**FOREIGN** **KEY** **(** regno **)**

**REFERENCES** Car **(**regno**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**)**

ENGINE **=** InnoDB**;**

1. Insert 100 rows into Passing

* Code for adding 100 rows

MariaDB **[**Oblig5**]>** INSERT INTO Passing SELECT DISTINCT regno**,**now**(),**3 FROM Car LIMIT 100**;**

Query OK**,** 100 rows affected **(**0**.**016 sec**)**

Records**:** 100 Duplicates**:** 0 Warnings**:** 0

## From dump

First we get the latest backup, and extract it.

Then after a quick inspection of the file we determine the insert values are between the lines 695 and 890 in the backup sql dump.

By writing a small command we extract only the relevant sql lines into their own sql file.

**sudo** sed -n -e "695,890p" "/extra/backup/DB/1582571328\_Backup\_2020-02-24\_14:08:48/extra/backup/DB/1582571328/Oblig5\_2020-02-24\_14:08:48.sql" **>** **/**home**/**admo**/**git**/**DAT151**/**recovered.sql

Then I source these commands in the sql database

SOURCE /home/admo/git/DAT151/recovered.sql

## Binary logs

During our testing we didn’t get the binary logs to show any of the changes that was done to the database. From searching we couldn’t find any relevant settings except adding

log\_bin = mysql-bin

to the mariadb config file. This at least enabled binary logs. But from our testing it still does not log the changes.

We will still explain how I would proceed if I would be able to see the changes in the file.

1. Flush the binary logs

MariaDB **[**Oblig5**]>** FLUSH LOGS;

1. Find the most relevant log

MariaDB **[**Oblig5**]>** SHOW MASTER STATUS;

1. Use mysqlbinlog to look through the log and find the last commit before dropping of the table
   1. This position will then be noted for later
   2. Command

$sudo mysqlbinlog **/**var**/**lib**/**mysql**/**mysql-bin.000001

1. Pipe the mysqlbinlog with a stop position into mysql.
   1. Command

mysqlbinlog **--**stop-position**=**TheFoundPosition **/**var**/**lib**/**mysql**/**mysql-bin.000001 **|** mysql -u Adrian -p Oblig5

## Results

From the results of our testing the script works and pulls all the generated binary files. Aswell as the sql dumps for each database. Where the biggest difficulty in the dump was to find the relevant data to re-insert. This could maybe have been easier if we had done a dump for each table and rather grouped the databases by themselves. Or if instead of looking for the relevant inserts we had just run the entire dump.

# Task 3: Replication