

MySQL maintenance plan

Alessandro Dotti Contra
alessandro@hyboria.org

July 20, 2011

Abstract

Many things can go wrong. But there is something that can be done to prevent, or at least reduce, the damages of a real disaster. This document describes yet another maintenance plan for MySQL. I currently use MySQL 4.1 on a Debian GNU/Linux system. Most of my databases uses the InnoDB engine.

Check databases integrity

The first thing to do before dumping the databases is to check their integrity. The following command:

```
$ mysqlcheck -u root --password=<password> \  
    --all-databases --analyze --check --optimize --auto-repair
```

performs tables analysis, checks for errors and optimize each database. Automatic repair will be performed if any table is find corrupted.

Full system dump

MySQL on Debian systems have binary logs enabled by default. There is a cron job which takes care of logs rotation and cleanup. Binary logs are kept enabled but automatic rotation will be disabled.

Edit the `/etc/mysql/debian-log-rotate` and set

```
KEEP_BINARY_LOG=0
```

Now issue the following commands:

```
$ echo "SET FOREIGN_KEY_CHECKS=0;" > mysql-dump  
$ mysqldump -u root --password=<password> \  
    --single-transaction --all-databases \  
    --flush-logs --master-data=2 --delete-master-logs >> mysql-dump  
$ echo "SET FOREIGN_KEY_CHECKS=1;" >> mysql-dump
```

The first and last commands disable foreign keys checks then enable them again. This is needed if you need to restore the system using the full dump. The `mysqldump` command performs the full system dump and creates a new binary log. The old logs are removed as they are now useless.

Databases dump

Now that a full system dump is available, it is useful to have a separate dump for each database.

```
$ echo "SET FOREIGN_KEY_CHECKS=0;" > [database]-dump  
$ echo "USE [database];" >> [database]-dump  
$ mysqldump -u root --password=<password> --opt [database] >> [database]-dump  
$ echo "SET FOREIGN_KEY_CHECKS=1;" >> [database]-dump
```

Replace `[database]` with the name of the database to be dumped.

System recovery

The following procedure describes how to perform a system recovery.
Use the full dump to restore the system:

```
$ mysql -u root --password <password> < mysql-dump
```

Now binary logs can be used to perform a finer grained recovery of the changes made to the system after the full dump.

```
$ mysqlbinlog logfile1 logfile2 ... | mysql -u root --password=<password>
```

or

```
$ mysqlbinlog logfile1 logfile2 ... > mysql-log  
$ mysql -u root --password=<password> < mysql-log
```

The second option is useful to examine the binary log before recovery.

It is also possible to perform a partial recovery of the binary log's content passing the `--start-datetime/--stop-datetime` and `--start-position/--stop-position` options to `mysqlbinlog`.

For single database recovery, the command is

```
$ mysql -u root --password=<password> < [database]-dump
```

Credits

Many thanks to Maurizio Lemmo for his useful explanations.