



TGM - HTBLuVA Wien XX  
Informationstechnologie

---

# Haidn & Siegel

## Backup with Mysql and Postgresql

---

Version	Autor	Datum	Status	Kommentar
0.1	Siegel	2014.11.28	Draft	
0.2	Haidn	2014.12.09	Draft	Mysql finished
0.3	Siegel	2014.12.14	Draft	Postgres finished
1.1	Haidn	2014.12.15	Finished	QA done

# Contents

<b>1</b>	<b>Working time</b>	<b>2</b>
1.1	Estimated . . . . .	2
1.2	Final . . . . .	3
<b>2</b>	<b>Working situation</b>	<b>4</b>
2.1	Database models . . . . .	4
2.2	Hannah . . . . .	4
2.3	Martin . . . . .	4
<b>3</b>	<b>Backups</b>	<b>5</b>
3.1	Why should backups be done? . . . . .	5
3.2	Logical versus physical backups . . . . .	5
3.3	Full versus incremental backups . . . . .	6
3.4	Online versus Offline backups . . . . .	6
<b>4</b>	<b>Mysql Backup</b>	<b>7</b>
4.1	Logical Backup into Files with mysqldump . . . . .	7
4.1.1	Performing a database dump . . . . .	7
4.1.2	Drop statements . . . . .	8
4.1.3	Performing a data restore . . . . .	8
4.2	Physical backup with mysqlbackup . . . . .	9
4.3	Restoring from another Database . . . . .	9
4.4	Physical backup using File system commands . . . . .	10
4.4.1	MyISAM . . . . .	10
4.4.2	InnoDB . . . . .	11
4.5	mysqlhotcopy . . . . .	11
4.5.1	Copying the files . . . . .	11
4.5.2	Restoring . . . . .	11
4.6	Backup of Triggers / Stored Routines . . . . .	12
4.7	Online Backup . . . . .	12
4.8	Remote Backups . . . . .	12
4.8.1	Using mysqldump to perform remote backups . . . . .	12
4.8.2	Using ftp to perform remote backups . . . . .	12
4.9	Automated Backups . . . . .	12
4.9.1	Backup at a certain time . . . . .	13
4.9.2	Using the backups . . . . .	13
<b>5</b>	<b>Postgres Backup</b>	<b>14</b>
5.1	Logical Backup into Files with pg_dump . . . . .	14
5.1.1	Performing a database dump with pg_dump [12] . . . . .	14
5.1.2	Pg_dump's Options . . . . .	15
5.1.3	Drop statements . . . . .	16
5.1.4	Performing a data restore . . . . .	16
5.2	Physical backups . . . . .	17
5.3	Backup of Triggers / Stored Routines . . . . .	17
5.4	Remote Backups . . . . .	17
5.5	Online Backups . . . . .	17
5.6	Automated Backups . . . . .	17
<b>6</b>	<b>Summary</b>	<b>18</b>

# 1 Working time

## 1.1 Estimated

Task	Person	Time in hours
Setting up the Databases	Haidn	0.5
	Siegel	0.5
Getting some informations about backups	Haidn	1
	Siegel	1
mysqldump	Haidn	1
	Siegel	1
mysqlbackup	Haidn	1
	Siegel	1
system-level commands	Haidn	1
	Siegel	1
mysqlhotcopy	Haidn	1
	Siegel	1
Automated Backup	Haidn	1
	Siegel	1
Documentation	Haidn	0.5
	Siegel	0.5
pg_dump	Haidn	2
	Siegel	2
pg_restore	Haidn	1
	Siegel	1
automated postgres	Haidn	2
	Siegel	2
Total	Haidn	13
	Siegel	12
<b>Total Team</b>		<b>25 hours</b>

## 1.2 Final

Task	Person	Time in hours
Setting up the Databases	Haidn	1
	Siegel	1
Getting some informations about backups	Haidn	2
	Siegel	1
mysqldump	Haidn	2
	Siegel	2
mysqlbackup	Haidn	0
	Siegel	2
system-level commands	Haidn	2
	Siegel	0
mysqlhotcopy	Haidn	1.5
	Siegel	1.5
Automated Backup	Haidn	1.5
	Siegel	2
Documentation Mysql	Haidn	1
	Siegel	2
pg_dump	Haidn	2
	Siegel	2
pg_restore	Haidn	0.5
	Siegel	0.5
Documentation Postgres	Haidn	1
	Siegel	1
Automated Postgres Backup	Haidn	1.5
	Siegel	1.5
Total	Haidn	16
	Siegel	16.5
<b>Total Team</b>		<b>32.5 hours</b>

Therefore it took us a little bit longer (about 10 hours more).

## 2 Working situation

### 2.1 Database models

We were using the create scripts from the last VSDB homework. In this we had done two similar create scripts using both mysql and postgres.

The database contains triggers and inserts for each table.

The ER can be seen in figures 1 and 2.

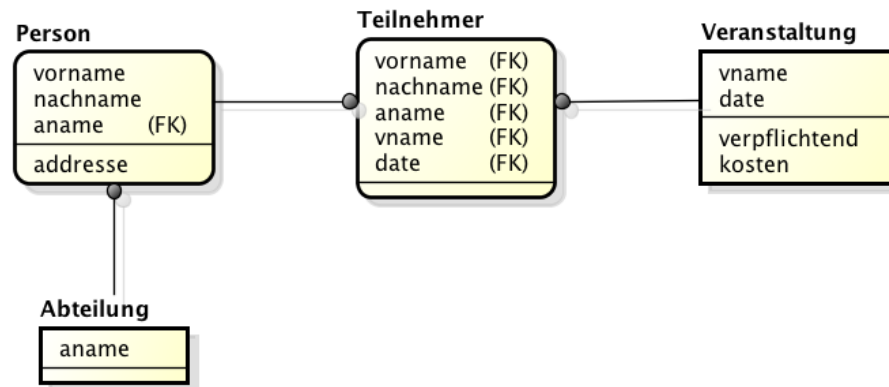


Figure 1: Entity Relationship Diagram for the mysql database

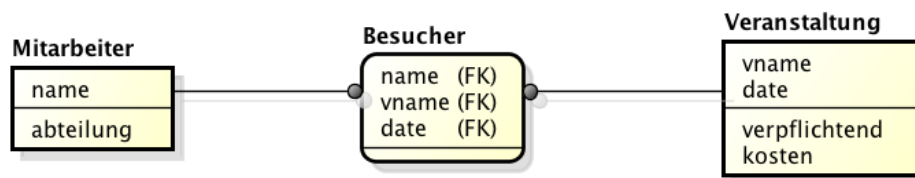


Figure 2: Entity Relationship Diagram for the postgres database

### 2.2 Hannah

Databases: Mysql Version 14.14, Distribution 5.5.40, PostgreSQL 9.3.5

DB-Server Environment (VM) : Ubuntu 14.04 LTS , it's IP: 192.168.117.131

### 2.3 Martin

Database: Mysql Version 14.14 Distribution 5.5.31

DB-Server Environment (VM) : Linux Debian 3.10.11-1

## 3 Backups

### 3.1 Why should backups be done?

After a drop-out, a recovery must be done.

To avoid a loss of data, a backup must be done before an drop-out occurs and it always should be current.

Backups are not only needed for recovery purposes, but also for archival storage purposes.

### 3.2 Logical versus physical backups

Logical backups save information represented as logical database structure (e.g. `Create Database`, `Create Table` statements) and content (e.g `Insert` statements). Physical backups consist of raw copies of the directories and files that store database contents.

Logical backups :

1. Backup is done by querying the [MySQL] server
2. Slower than physical methods
3. Output is larger than physical methods
4. The Backup and the Restore can be done either for all databases (server level), only for one database (database level) or only for specific tables (table level)
5. Doesn't include log or config files
6. Backups are mostly done with the database still running
7. Can be easily imported
8. Backups stored in logical format are machine independent!

[3]

Physical backups :

1. Exact copy of database files which are stored on the disk
2. Output is more compact than logical ones
3. The granularity of the data that can be stored depends on the engine (e.g. InnoDB shares files with other tables..)
4. Can include log or config files
5. Backups stored in logical format are machine dependent!
6. Backups are seldom done with the database running, and if then the database files must be locked.

[3]

### 3.3 Full versus incremental backups

"Some file system implementations enable "snapshots" to be taken. These provide logical copies of the file system at a given point in time, without requiring a physical copy of the entire file system. [...] MySQL itself does not provide the capability for taking file system snapshots. It is available through third-party solutions such as Veritas, LVM, or ZFS.", [3]

"A full backup includes all data managed by a MySQL server at a given point in time. An incremental backup consists of the changes made to the data during a given time span (from one point in time to another). MySQL has different ways to perform full backups, such as those described earlier in this section. Incremental backups are made possible by enabling the server's binary log, which the server uses to record data changes.

Incremental recovery is recovery of changes made during a given time span. This is also called point-in-time recovery because it makes a server's state current up to a given time. Point-in-time recovery is based on the binary log and typically follows a full recovery from the backup files that restores the server to its state when the backup was made. Then the data changes written in the binary log files are applied as incremental recovery to redo data modifications and bring the server up to the desired point in time.", [3]

### 3.4 Online versus Offline backups

Online backups, also called hot backups, take place while the server is running so that the database can still be used.

Offline backups, also called cold backups, take place while the server is not running and therefore the database is not available.

Online backups :

1. Clients can still access the database
2. The backup must be made carefully, in order to secure, that the clients have not changed information in the mean time that could compromise the backup's integrity

[3]

Offline backups :

1. Clients can not access the database
2. The backup is easier

A similar distinction between online and offline applies for recovery operations. [3]

## 4 Mysql Backup

### 4.1 Logical Backup into Files with mysqldump

Mysql has an option called mysqldump. Mysql dump can connect to local or remote servers.[3]

#### 4.1.1 Performing a database dump

To do a backup from only one database, the following command needs to be executed:

```
mysqldump -u [user] -p [database_name] > dumpfilename.sql
```

After typing in the password, a file will be available which, in this case is called `dumpfilename.sql`.

The content of the dumpfile is the following:

---

```
-- MySQL dump 10.13 Distrib 5.5.40, for debian-linux-gnu (x86_64)
--
-- Host: localhost  Database: insyl
--
-- Server version      5.5.40-Oubuntu0.14.04.1

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table 'Abteilung'
--

DROP TABLE IF EXISTS 'Abteilung';
/*!40101 SET @saved_cs_client = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE 'Abteilung' (
  'aname' varchar(255) NOT NULL,
  'sync_state' enum('current','old','new','syncing','deleting') NOT NULL DEFAULT 'new',
  PRIMARY KEY ('aname','sync_state')
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;
...
```

---

Even though that in this case there is only one create table command, there is a lot of bulk.

In the next example, the Inserts into a table can be seen, and here it actually is not too much bulk. Still, whenever saving a database dump like this, the file will be quite big, which might be a disadvantage!

---

```
INSERT INTO 'Person' VALUES
('Aly','Ahmed','Facility Management','Doppelte gasse','current'),
('Dominik','Scholz','IT','Schwarze gasse','current'),
('Elias','Frantar','Kindergarten','Heiligenstadt gasse','current'),
('Hannah','Siegel','HR','Max Kahrer gasse','current'),
('Jakob','Saxinger','Kueche','Max Soundso gasse','current'),
('Martin','Haidn','Managment','Gruene gasse','current'),...
```

---



#### Backup of only one table using mysqldump

```
mysqldump -u [user] -p [database_name] [table_name] > dumpfilename.sql
```

#### Backup of more than one database using mysqldump

```
mysqldump -u [user] -p --databases [database_name1] [database_name2] > dumpfilename.sql
```

#### Backup of all databases using mysqldump

```
mysqldump -u [user] -p --all-databases > dumpfilename.sql
```

#### Backup of only the structure without any data

```
mysqldump -u [user] -p [-d|--no-data] [database_name] > dumpfilename.sql
```

### 4.1.2 Drop statements

If drop-statements should be added, the following parameters can simply be added: [2]

Format	Description
<code>--add-drop-database</code>	Adds a DROP DATABASE statement before each CREATE DATABASE statement
<code>--add-drop-table</code>	Adds a DROP TABLE statement before each CREATE TABLE statement

The option `--add-drop-trigger` was supported in version 5.1, but it is not available anymore in 5.5:

---

```
root@ubuntu:/var/lib/mysql# mysqldump -u root -p insy2 --add-drop-trigger > df1.sql
mysqldump: unknown option '--add-drop-trigger'
```

---

Weirdly, even though we didn't add the `--add-drop-table` command in the first place, the dump includes the DROP TABLE IF EXISTS Command, as it can be seen in the first code snippet under this section:

---

```
DROP TABLE IF EXISTS 'Abteilung';
```

---

### 4.1.3 Performing a data restore

After having dropped the database, the following command was restoring the data. This has worked out fine: To do a backup from only one database, the following command needs to be executed:

```
mysql -u root -p [database_name] <dumpfilename.sql
```

With this type of recovery, even the triggers have been imported.

## 4.2 Physical backup with mysqlbackup

A physical backup tool is available in the mysql Enterprise Edition. These information could be found under [4]. Of course, it might not be the most trustful source, because mysql will surely try to promote it's product, but still it might be about that much faster:

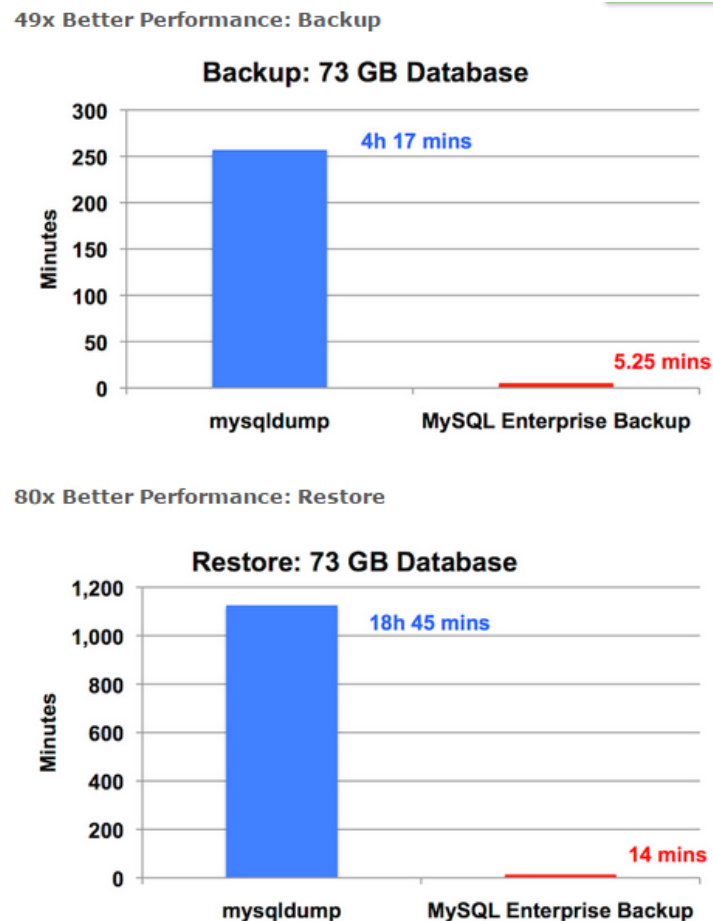


Figure 3: Difference between mysqlbackup and mysqldump

## 4.3 Restoring from another Database

Also, a database could be copied from another using these commands. We have tried this out an it worked.[5]

---

```
MyISAM:
CREATE TABLE db2.mytable LIKE db1.mytable;
ALTER TABLE db2.mytable DISABLE KEYS;
INSERT INTO db2.mytable SELECT * FROM db1.mytable;
ALTER TABLE db2.mytable ENABLE KEYS;
```

---

```
INNODB:
CREATE TABLE db2.mytable LIKE db1.mytable;
```

```
INSERT INTO db2.mytable SELECT * FROM db1.mytable;
```

---

## 4.4 Physical backup using File system commands

In order to copy the files, the location must be found out. This can be done with the `select @@datadir` command within mysql.

```
mysql> select @@datadir;
+-----+
| @@datadir |
+-----+
| /var/lib/mysql/ |
+-----+
1 row in set (0.00 sec)
```

Figure 4: Output of the `select @@datadir` command

In figure 9, the content of the `insy1` database can be seen. The only MyISAM table is 'Logged'. Also, each trigger has its own file.

```
root@ubuntu:/var/lib/mysql/insy1# ls
Abteilung.frm      insertperson.TRN      Teilnehmer.frm
Abteilung.TRG      insertteilnehmer.TRN  Teilnehmer.TRG
db.opt             insertveranstaltung.TRN  updateabteilung.TRN
deleteabteilung.TRN  Logged.frm             updateperson.TRN
deleteperson.TRN    Logged.MYD              updateteilnehmer.TRN
deleteteilnehmer.TRN  Logged.MYI              updateveranstaltung.TRN
deleteveranstaltung.TRN  Person.frm              Veranstaltung.frm
insertabteilung.TRN    Person.TRG               Veranstaltung.TRG
```

Figure 5: Content of the `var/etc/mysql` folder

### 4.4.1 MyISAM

Copying files when using a MyISAM Database, is possible, because every Table maps to exactly one file.

”However, you cannot just move the `.frm`. You must move all components.”, [5]

There are three files (see also in figure 9) which have something to do with the table:

- `/var/lib/mysql/insy1/Logged.frm`
- `/var/lib/mysql/insy1/Logged.MYD` (Table Database)
- `/var/lib/mysql/insy1/Logged.MYI` (Table Indexes)

When simply coping these files, dropping the table and coping them back, and then calling a `Select * from Logged` command, the following error message occurs:

---

```
ERROR 1017 (HY000): Can't find file: './insy1/Logged.frm' (errno: 13)
```

---

When searching for a solution in the web, mostly it says that it is really better backing mysql databases up with `mysqlhotcopy`.

#### 4.4.2 InnoDB

On the other hand, doing a backup when coping files, is "risky (near suicidal) with InnoDB.",[5] Therefore, whenever using InnoDB, a backup should be done with `mysqldump`.

### 4.5 mysqlhotcopy

"mysqlhotcopy is a Perl script that was originally written and contributed by Tim Bunce. It uses FLUSH TABLES, LOCK TABLES, and cp or scp to make a database backup. It is a fast way to make a backup of the database or single tables, but it can be run only on the same machine where the database directories are located. mysqlhotcopy works only for backing up MyISAM and ARCHIVE tables. It runs on Unix.",[6]

#### 4.5.1 Copying the files

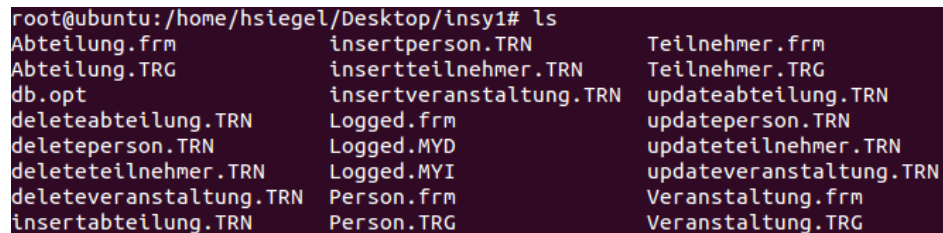
Using the mysqlhotcopy is a really easy way to copy the files. In the following example, it can be seen, what command must be used and the output.

---

```
hsiegel@ubuntu:~/Documents/sync_het_db/Create$ sudo /usr/bin/mysqlhotcopy -u root -p
secret_password insy1 /home/hsiegel/Desktop --allowold --keepold
Flushed 5 tables with read lock ('insy1'. 'Abteilung', 'insy1'. 'Logged', 'insy1'. 'Person',
'insy1'. 'Teilnehmer', 'insy1'. 'Veranstaltung') in 0 seconds.
Locked 0 views () in 0 seconds.
Copying 24 files...
Copying indices for 0 files...
Unlocked tables.
mysqlhotcopy copied 5 tables (24 files) in 0 seconds (0 seconds overall).
```

---

This simply copies the files from `/var/lib/mysql/insy1` to the source destination.



```
root@ubuntu:/home/hsiegel/Desktop/insy1# ls
Abteilung.frm      insertperson.TRN    Teilnehmer.frm
Abteilung.TRG      insertteilnehmer.TRN Teilnehmer.TRG
db.opt             insertveranstaltung.TRN updateabteilung.TRN
deleteabteilung.TRN Logged.frm           updateperson.TRN
deleteperson.TRN   Logged.MYD          updateteilnehmer.TRN
deleteteilnehmer.TRN Logged.MYI          updateveranstaltung.TRN
deleteveranstaltung.TRN Person.frm           Veranstaltung.frm
insertabteilung.TRN Person.TRG          Veranstaltung.TRG
```

Figure 6: Content of the insy1 folder, which has been generated using mysqlhotcopy

#### 4.5.2 Restoring

"To restore the backup from the mysqlhotcopy backup, simply copy the files from the backup directory to the `/var/lib/mysql/db-name` directory. Just to be on the safe-side, make sure to stop the mysql before you restore (copy) the files. After you copy the files to the `/var/lib/mysql/db-name` start the mysql again.",[7] In our example, restoring after having done a mysqlhotcopy didn't work either, with the same error message as before:

---

```
ERROR 1017 (HY000): Can't find file: './insy1/Logged.frm' (errno: 13)
```

---

## 4.6 Backup of Triggers / Stored Routines

When using mysqldump, the triggers have been saved automatically and the import has not been a problem.

Because the mysqlhotcopy didn't work, we couldn't evaluate it by our selves if the triggers were restored. But normally, it should be possible using mysqlhotcopy as well.

## 4.7 Online Backup

With MySQL Enterprise Backup, a hot backup is possible.

Also, mysqlhotcopy, as the name already says, is also able to perform hot backups, because the Perl script handles the locking. Mysqldump is executing Select statements on the database, therefore it is a hot copy as well, but in this case, it might be really slow for users and the task itself.

## 4.8 Remote Backups

### 4.8.1 Using mysqldump to perform remote backups

You can specify the server name as an option to mysqldump: `mysqldump --host [server_name] [database_name] > dumpfilename.sql`

### 4.8.2 Using ftp to perform remote backups

As described in section 4.4, the files can simply be copied, and this process of copying can also be performed using ftp.

## 4.9 Automated Backups

An important way of performing backups is that they can also be automatised. This has the advantage, that the administrator doesn't has to think about doing a backup and therefore is not able to forget it, and also backups can then be scheduled at times, when the database server might not be very busy, so the backup is faster and the client do not notice that a backup task has just been performed.

Automised Backups with mysql are quite easy. This can be done using the tool automysqlbackup.

The following commands have to be done:

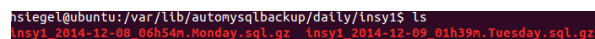
```
sudo apt-get install automysqlbackup
sudo automysqlbackup
```

The main configuration file for automysqlbackup is located at `/etc/default/automysqlbackup`.

"If you only want to back up certain databases, you can specify them in the DBNAMES configuration variable. Conversely, if you want to backup everything except certain databases, you can use the DBEXCLUDE configuration variable to list what to exclude.",[9]

The backups will be located at `/var/lib/automysqlbackup`. There are three folders: **daily**, **weekly**, **monthly**. The automatised backup worked, in my daily folder, each day there is a new File.

"By default, the daily folder will contain all of the last seven days. The weekly folder will grow to contain the database as it was on Sunday each of the last fifty-two weeks. Similarly, monthly will contain the end of all each of the last twelve months.",[9]



```
inslegel@ubuntu: /var/lib/automysqlbackup/daily/insy1$ ls
insy1_2014-12-08_06h54m.Monday.sql.gz  insy1_2014-12-09_01h39m.Tuesday.sql.gz
```

Figure 7: Content of the `/var/lib/automysqlbackup` folder with two automatised backups

#### 4.9.1 Backup at a certain time

Because we couldn't find the possibility to set a time when the backup should be done with the `automysqlbackup` option, we looked further.

There is the possibility to use the cron-daemon. This is a program which can execute shell scripts automatically. All that had to be done, was adding a line into the cron-table.

This can be done using the `crontab -e` command.

In figure 8 the crontab syntax is described. We added the following line:

---

```
15 20 * * * /usr/sbin/automysqlbackup
```

---

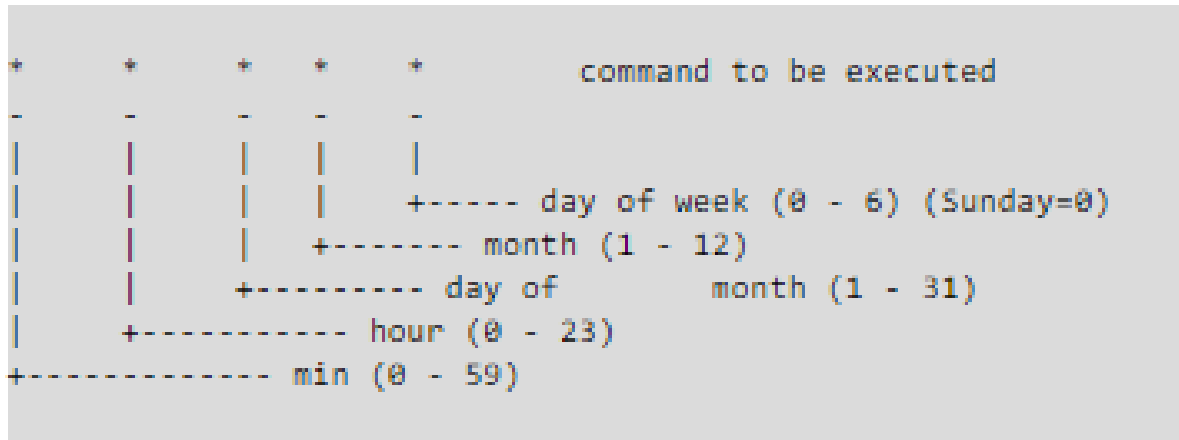


Figure 8: Usage of the crontab file[11]

ausführlicher

#### 4.9.2 Using the backups

"The `.gz` extension means it's compressed. To restore a database, you'd first have to uncompress the dump: `gunzip filename.sql.gz`",[10]. The unzipped file will look exactly the same like an sql file which has been generated using `mysqldump`. It can then be restored.

## 5 Postgres Backup

### 5.1 Logical Backup into Files with pg\_dump

Postgres comes with an option called pg\_dump.

#### 5.1.1 Performing a database dump with pg\_dump [12]

To do a backup from only one database, the following command needs to be executed:

```
pg_dump -U [user] [database_name] -f dumpfilename.sql
```

Then a file will be available which, in this case, is called dumpfilename.sql.

The content of the dumpfile is the following:

---

```
--
-- PostgreSQL database dump
--
SET statement_timeout = 0;
SET lock_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SET check_function_bodies = false;
SET client_min_messages = warning;
--
-- Name: plpgsql; Type: EXTENSION; Schema: -; Owner:
--
CREATE EXTENSION IF NOT EXISTS plpgsql WITH SCHEMA pg_catalog;

--
-- Name: EXTENSION plpgsql; Type: COMMENT; Schema: -; Owner:
--
COMMENT ON EXTENSION plpgsql IS 'PL/pgSQL procedural language';

SET search_path = public, pg_catalog;
--
-- Name: action_type; Type: TYPE; Schema: public; Owner: hsiegel
--

CREATE TYPE action_type AS ENUM (
    'insert',
    'update',
    'delete'
);
.....

CREATE TABLE logged (
    id integer NOT NULL,
    action action_type,
    tablename character varying(255) NOT NULL,
    old_values character varying(255) NOT NULL,
    new_values character varying(500),
    date_done timestamp without time zone
);

.....
```

---

Same as with Mysql, one create table command is producing a lot of lines.

In the next example, the Inserts into a table can be seen, and here it actually is quite short.

---

```

COPY mitarbeiter (name, abteilung, sync) FROM stdin;
Hannah Siegel HR      current
Nikolaus Schrack  Analysten      current
Paul Adeyemi  Sportabteilung current
Wolfram Soyka  Finance current
Jakob Saxinger Sales  current
Philip Schwarzkopf Sales  current
Elias Frantar  IT      current
Gary Ye  IT      current
Aly Ahmed  Facility Management  current
Martin Haidn  Managment      current
Dominik Scholz IT      current

```

---

### Backup of all databases using pg\_dump

```
pg_dumpall -U [user] -f dumpfilename.sql
```

### Backup of only one table using pg\_dump

```
pg_dump -U [user] -f dumpfilename.sql -t mitarbeiter
```

## 5.1.2 Pg\_dump's Options

### –data-only

Dump only the data, not the schema (data definitions).

### –create

Adds the line:

---

```

CREATE DATABASE vsdb_03 WITH TEMPLATE = template0 ENCODING = 'UTF8' LC_COLLATE =
    'en_US.UTF-8' LC_CTYPE = 'en_US.UTF-8';

```

---

### –F format

*p plain*

Output a plain-text SQL script file (the default).

*c custom*

Output a custom archive suitable for input into pg\_restore. This is the most flexible format in that it allows reordering of loading data as well as object definitions. This format is also compressed by default.

The output is an .sql File but it is not readable:

Figure 9: Content of the file

*t tar*

Output a tar archive suitable for input into pg\_restore. Using this archive format allows reordering



and/or exclusion of database objects at the time the database is restored. It is also possible to limit which data is reloaded at restore time.

#### **-inserts**

"Dump data as INSERT commands with explicit column names (INSERT INTO table (column, ...) VALUES ...). This will make restoration very slow; it is mainly useful for making dumps that can be loaded into non-PostgreSQL databases. Also, since this option generates a separate command for each row, an error in reloading a row causes only that row to be lost rather than the entire table contents."  
[12]

This generates the following output:

---

```
INSERT INTO mitarbeiter VALUES ('Hannah Siegel', 'HR', 'current');
INSERT INTO mitarbeiter VALUES ('Nikolaus Schrack', 'Analysten', 'current');
[...]
```

---

```
INSERT INTO mitarbeiter VALUES ('Aly Ahmed', 'Facility Management', 'current');
INSERT INTO mitarbeiter VALUES ('Martin Haidn', 'Managment', 'current');
INSERT INTO mitarbeiter VALUES ('Dominik Scholz', 'IT', 'current');
```

---

#### **-t [table\_name]**

This option dumps only one specific table.

### **5.1.3 Drop statements**

If the option `--clean` was added to the `pg_dump` command, the triggers, tables and so on will be dropped:

---

```
DROP TRIGGER update_veranstaltung_trigger ON public.veranstaltung;
[...]
```

---

```
ALTER TABLE ONLY public.veranstaltung DROP CONSTRAINT pk_veranstaltung;
[...]
```

---

```
DROP TABLE public.veranstaltung;
[...]
```

---

```
DROP SEQUENCE public.logged_id_seq;
[...]
```

---

```
DROP FUNCTION public.update_veranstaltung();
[...]
```

---

```
DROP TYPE public.sync_type;
[...]
```

---

```
DROP EXTENSION plpgsql;
DROP SCHEMA public;
```

---

### **5.1.4 Performing a data restore**

Restoring from the sql file using `pg_restore` didnt work at first, this was the error message:

---

```
pg\_restore: [archiver] input file appears to be a text format dump. Please use psql.
```

---

Then we tried it with the file which we had generated using the `-F custom` parameter from `pg_dump`. There was no exception or anything, but somehow the import was not succesfull.

Happily, the restore worked with the normal sql file. All that had to be done was importing in when we were logged in in the database using the `\i filename` command.

## 5.2 Physical backups

In the Postgres Manual the following command could be found:

```
tar -cf backup.tar /usr/local/pgsql/data
```

But in our case we were not able to find this path. It just didn't exist.

"The database server must be shut down in order to get a usable backup. [...] Restoring individual tables or databases from their respective files or directories will not work because the information contained in these files is not usable without the commit log files, pg\_clog/\*, which contain the commit status of all transactions.",[16]

"An alternative file-system backup approach is to make a "consistent snapshot" of the data directory, if the file system supports that functionality (and you are willing to trust that it is implemented correctly). The typical procedure is to make a "frozen snapshot" of the volume containing the database, then copy the whole data directory (not just parts, see above) from the snapshot to a backup device, then release the frozen snapshot. This will work even while the database server is running. However, a backup created in this way saves the database files in a state as if the database server was not properly shut down; therefore, when you start the database server on the backed-up data, it will think the previous server instance crashed and will replay the WAL log. This is not a problem; just be aware of it (and be sure to include the WAL files in your backup). You can perform a CHECKPOINT before taking the snapshot to reduce recovery time.",[16]

Other references were leading to the PGDATA folder. This folder is located at `/var/lib/postgresql/9.3/main` in our case. Still, we were not able to locate the relevant files. `Pg_dump` therefore is still the better backup method in our opinion.

## 5.3 Backup of Triggers / Stored Routines

With the `pg_dump` command, triggers and functions had been restored as well.

## 5.4 Remote Backups

The `pg_dump` method has a parameter `-h` in which an host can be specified. Therefore it enables a remote host backup.

## 5.5 Online Backups

The `pg_dump` method enables online backups. With an physical method, the database server must be shut down in order to get a usable backup.

## 5.6 Automated Backups

First, we downloaded an `.sh` file from [15]. It basically is the same file as the `automysqlbackup`, but customised for PostgreSQL.

We put the file into the `/home/hsiegel/Documents/sync_het_db` folder and we added the following line to our crontab (using the `crontab -e` command, just as before).

---

```
40 15 * * * /home/hsiegel/Documents/sync_het_db/autopgbackup.sh
```

---

In order to get this working, we had to change the Database names in the script from `all` to the ones we wanted to backup, because otherwise it wouldn't work. As it can be seen in figure 10, the backup happened at exactly 15:40. The database `vsdb_03` has been successfully backedup (figure 11).

```

hsiegel@ubuntu: ~/Documents/sync_het_db/backup
hsiegel@ubuntu:~/Documents/sync_het_db/backup$ date
Tue Dec 16 15:38:14 CET 2014
hsiegel@ubuntu:~/Documents/sync_het_db/backup$ ls
hsiegel@ubuntu:~/Documents/sync_het_db/backup$ date
Tue Dec 16 15:40:05 CET 2014
hsiegel@ubuntu:~/Documents/sync_het_db/backup$ ls
daily monthly weekly
hsiegel@ubuntu:~/Documents/sync_het_db/backup$

```

Figure 10: Time of the backup

```

hsiegel@ubuntu:~/Documents/sync_het_db/backup/daily/vsdb_03$ ls
vsdb_03_2014-12-16.Tuesday.sql.gz

```

Figure 11: Content of the backup folder

## 6 Summary

After having tried out all the possible backup methods, we finally can conclude, that the `mysql_dump` and the `pg_dump` were the best tools.

This was because the logical files were better readable, and it was easier doing the backup. The two tools provide the user with a wide range of possibilities.

Automated backups should be done using the cron-deamon. We liked this option, because it was really easy and it also provides a lot of functionality. Besides this, the crontab was a centralized place in which we had both the mysql and the postgres backup and therefore it would be easy to change both of them at the same time.

## Problems

### **Mysql connection didn't work out**

When trying to connect to the mysql Database, the following error was thrown: **ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/var/run/mysqld/mysqld.sock'**

The following steps had to be done:

1. Super user or sudo should be used. (`sudo su`)
2. The file `/etc/mysql/my.cnf` must be opened
3. The bind address had to be changed to `127.0.0.1`
4. Then `service mysql restart` must be executed

[1]

<http://www.pgadmin.org/docs/1.8/pgagent-install.html>

## References

- [1] **ERROR 2002: Can't connect to local MySQL server through socket StackOverflow**  
Peter Mortensen Apr 13 at 10:45 and rshahriar Oct 11 '12 at 6:27  
<http://stackoverflow.com/questions/11657829/error-2002-hy000-cant-connect-to-local-mysql-server-through-socket-var-run>  
last used: 2014.12.05, 13:15
- [2] **Mysql Manual 5.5**, mysqldump — A Database Backup Program  
<http://dev.mysql.com/doc/refman/5.5/en/mysqldump.html>  
last used: 2014.12.05, 13:55
- [3] **Mysql Manual 5.5** Backup and Recovery Types  
<http://dev.mysql.com/doc/refman/5.5/en/backup-types.html>  
last used: 2014.12.06, 19:01
- [4] **Mysql Website**MySQL Enterprise Backup  
<http://www.mysql.com/products/enterprise/backup.html>  
last used: 2014.12.08, 10:49
- [5] **StackOverflow**, RolandoMySQLDBA - answered Mar 7 '12 at 17:45 St  
<http://serverfault.com/questions/367255/linux-mysql-is-it-safe-to-copy-mysql-db-files-with-cp-command-from-one-db-to>  
last used: 2014.12.08, 11:39
- [6] **Mysql Manual 5.5** mysqlhotcopy — A Database Backup Program  
<http://dev.mysql.com/doc/refman/5.5/en/mysqlhotcopy.html>  
last used: 2014.12.08, 14:16
- [7] **Backup and Restore MySQL Database using mysqlhotcopy**, Ramesh Natarajan  
<http://www.thegeekstuff.com/2008/07/backup-and-restore-mysql-database-using-mysqlhotcopy/>  
last used: 2014.12.08, 14:44
- [8] **How To Backup MySQL Databases on an Ubuntu VPS**, Justin Ellingwood  
<https://www.digitalocean.com/community/tutorials/how-to-backup-mysql-databases-on-an-ubuntu-vps>  
last used: 2014.12.08, 15:30
- [9] **Scheduled DB backup with automysqlbackup**, January 5, 2013  
<https://pelanne.com/blog/scheduled-db-backup-automysqlbackup/>  
last used: 2014.12.10, 09:33
- [10] **Creating MySQL Backups With AutoMySQLBackup On Ubuntu 9.10**  
Falko Timme, January 27, 2010  
<http://www.howtoforge.com/creating-mysql-backups-with-automysqlbackup-on-ubuntu-9.10>  
last used: 2014.12.10, 09:35
- [11] **Crontab – Quick Reference** Hemant Sharma  
<http://www.adminschoice.com/crontab-quick-reference>  
last used: 2014.12.14, 15:59
- [12] **Postgres Documentation** SQL Dump  
<http://www.postgresql.org/docs/9.1/static/backup-dump.html>  
last used: 2014.12.14, 19:32

- [13] **Postgres Documentation** pg\_restore  
*<http://www.postgresql.org/docs/9.1/static/app-pgrestore.html>*  
last used: 2014.12.14, 20:03
- [14] **bjoerne.com** 4. September 2013  
*<http://www.bjoerne.com/mysql-backups-mit-automysqlbackup-erstellen/>*  
last used: 2014.12.15, 15:16
- [15] **Autobackup Postgres Script** Aaron Axelsen, 2005 4. September 2013  
*<http://autopgsqlbackup.frozenpc.net>*  
last used: 2014.12.15, 15:17
- [16] **PostgreSQL 9.3 Documentation** Chapter 24. Backup and Restore  
*<http://www.postgresql.org/docs/9.3/static/backup-file.html>*  
last used: 2014.12.15, 16:39