Kapitel 4

Backup and Recovery

Datenbankbackups werden erstellt, damit beschädigte Datenbanken wiederhergestellt werden können. Das Sichern und Wiederherstellen von Daten muss jedoch auf die entsprechende Umgebung und auf die verfügbaren Ressourcen abgestimmt werden. Ein wichtiger Punkt bei der Wahl der richtigen Backupstrategie ist natürlich die Verhältnismäßigkeit zwischen den Anforderungen, die an das Recovery gestellt werden und den mit dem geplanten Backupkonzept verbundenen Kosten, Ressourcen und sonstigen Aufwand.

Folgende Punkte spielen eine Rolle bei der Entscheidung für eine Backupstrategie:

- Wie lange darf die Wiederherstellung dauern?
- Kann die Datenbank offline gesichert werden? Häufig ist dies nur nachts oder am Wochenende möglich. Muss die Datenbank 7x24-Stunden in der Woche zur Verfügung stehen, so kann das Backup nur online erfolgen. Bei vielen Produktiv-Systemen gibt es jedoch vereinbarte Wartungsfenster, in denen ein Offline-Backup durchgeführt werden kann.
- Wie groß ist das zu erwartende Datenvolumen? Bei sehr großen Datenbanken müssen durchsatzstarke und entsprechend dimensionierte Sicherungsmedien und eine leistungsstarke Hardware (Bandlaufwerk, Netzwerk, CPU, Plattenkontroller, etc.) zum Einsatz kommen.

4.1 Backuptypen

Grundsätzlich wird zwischen zwei Backuparten unterschieden, dem physischen und dem logischen Backup. Beim physischen Ansatz werden die Daten von den Platten blockweise auf ein Sicherungsmedium kopiert (Binary Backup). Bei einem logischen Backup werden die logische Struktur und der Inhalt von Datenbank-Objekten in eine Datei geschrieben.

Werden physische Backups bei laufender Datenbank angefertigt, sind sie zwangsläufig nicht konsistent. Um bei der Wiederherstellung zu einem konsistenten Stand zu gelangen, muss der Datenbank bekannt sein, welche Änderungen während und nach dem Backup angefallen sind. Diese Information speichern Datenbanken üblicherweise in Logdateien, die ebenfalls archiviert werden. Bei der Wiederherstellung werden die archivierten Logdateien eingespielt, wodurch letztendlich ein konsistenter Stand erreicht wird.

Folgende, weitere Startegien bei Backups werden unterschieden, wobei hier unterschiedliche Implementierungen der Hersteller zu berücksichtigen sind:

- Online vs. Offline Backup
- Local vs. Remote Backup
- Full vs. Incremental Backup

4.2 MySQL bzw. MariaDB

In MySQL haben Sie die Möglichkeit, einzelne Tabellen mit Daten, aber auch komplette Datenbanken mit den enthaltenen Tabellen bei laufendem Betrieb des MySQL Servers zu sichern. Eine vollständige Datensicherung ist aber nur gewährleistet, wenn zum Zeitpunkt des Backups keine Schreibzugriffe auf die zu sichernden Datenbanken bzw. auf die Tabellen der zu sichernden Datenbanken erfolgen.

Um eine Datensicherung bei laufendem Betrieb des MySQL-Servers durchzuführen, beachten Sie die folgenden Punkte:

- Stellen Sie sicher, dass keine Schreibzugriffe auf die zu sichernden Tabellen erfolgen, indem Sie die entsprechenden Tabellen sperren.
- Leeren Sie alle internen Caches aller während des Zeitpunkts der Datensicherung geöffneten Tabellen.

4.2.1 Tabellen sperren

Um sicherzustellen, dass während der Datensicherung keine Schreibzugriffe auf die geöffneten Tabellen erfolgen, müssen die Tabellen gesperrt werden.

Die Sperrung ist dabei in mehreren Stufen möglich:

- Die angegebenen Tabellen werden mit einem Schreibschutz versehen, Leseoperationen sind aber zugelassen (`READ`)
- Die SQL-Anweisung `INSERT` ist zugelassen, solange keine Konflikte mit dem Schreibschutz auftreten (`READ LOCAL`)
  - im Standardspeicherformat InnoDB bei MySQL-DBs besteht kein Unterschied zwischen `READ` und `READ LOCAL`.
- Die angegebenen Tabellen sind für alle Benutzer außer dem, welcher die Sperrung durchführt, für Schreib- und Lesezugriffe gesperrt (`WRITE`).

Für die Sperrung von Tabellen können Sie die SQL-Anweisung `LOCK TABLE` verwenden.

SQL-Syntax:

```
LOCK TABLE <Tabellenname> {READ | READ LOCAL | [LOW PRIORITY] WRITE}
[,<Tabellenname> {READ | READ LOCAL| WRITE}]
```

Um die Sperre einer Tabelle wieder aufzuheben, benutzen Sie die SQL-Anweisung `UNLOCK TABLES`, die alle Tabellensperrungen eines Benutzers wieder aufhebt.

4.2.2 Interne Caches leeren

Nach dem Sperren der notwendigen Tabellen, leeren Sie alle internen Caches bzw. die Caches der zu sichernden Tabellen, welche MySQL benutzt (`FLUSH TABLES`). Mit dieser Anweisung werden alle geöffneten Tabellen geschlossen, auch wenn diese benutzt werden.

Bsp.:

```
use fitnesscenter;
flush tables mitarbeiter, trainer;
```

bewirkt Folgendes:

- Schließen der beiden Tabellen
- das Ablegen aller für diese Tabellen im Speicher befindlichen Daten in die entsprechenden Dateien.

4.2.3 Möglichkeiten zur Durchführung der Datensicherung in MySQL

- Erstellung von Datenbankkopien auf Betriebssystemebene
- Sicherung von Tabellen und der Inhalt mit der SQL-Anweisung `SELECT INTO OUTFILE` in der MySQL-Konsole
- Verwendung des Dienstprogramms `mysqldump`
- Erstellung von Sicherungsdateien mithilfe eines grafischen Benutzerclients wie bspw. `phpmyadmin`

4.3 Datensicherung durchführen

4.3.1 Betriebssystemebene

Jede MySQL-DB findet man in der Verzeichnisstruktur des Betriebssystems als eigenes Verzeichnis in data der MySQL Installlation, daher brauchen nur diese Dateien bzw. Verzeichnisse kopiert werden. Beachten Sie bei der Sicherung auf OS-Ebene folgende Punkte:

- Sperren und leeren der Caches
  - wenn alle Datenbanken geflusht und gesperrt werden sollen, kann folgender Befehl genutzt werden:

    ```
    FLUSH TABLES WITH READ LOCK;
    ```

- Kopieren sie die DB einschließlich der Tabellendefinitionen
- Sichern Sie das Zugriffs- und Berechtigungssystem (DB: mysql)

4.3.2 SELECT INTO OUTFILE

Mit der SQL-Anweisung `SELECT INTO OUTFILE` können Sie Daten einer MySQL-Tabelle in eine delimited ASCII-Datei sichern. Bei der Datensicherung mithilfe dieser Anweisung geht die Tabellenstruktur verloren, da nur die Datensätze in die anegebene ASCII-Datei abgespeichert werden.

Bsp:

```
use fitnesscenter;
SELECT <Auswahl> INTO OUTFILE '<ASCII-Dateiname>' [Optionen] FROM <Tabellenname>;
```

Die vollständige Syntax lautet wie folgt:

```
SELECT ... INTO OUTFILE 'file_name'
       [CHARACTER SET charset_name]
       [export_options]

export_options:
       [{FIELDS | COLUMNS}
              [TERMINATED BY 'string']
              [[OPTIONALLY] ENCLOSED BY 'char']
              [ESCAPED BY 'char']
       ]
       [LINES
              [STARTING BY 'string']
              [TERMINATED BY 'string']
       ]
```

4.3.3 Daten mit mysqldump sichern

Mit dem Dienstprogramm `mysqldump` können Sie logische Backups von Datenbanken erstellen (Definitionen von Datenbankobjekten wie Tabellen, Stored Procedures o.ä. sowie Sicherung der eigentlichen Daten). Der Vorteil hierbei liegt unter anderem auch darin, dass die Datenbanken auf einen anderen SQL-Server umgezogen werden können.Mysqldump bietet eine Vielzahl an Optionen, die hier nicht aufgelistet werden sollen, sondern diese können unter https://dev.mysql.com/doc/refman/8.0/en/mysqldump.html eingesehen werden (wichtige Optionen: `--opt, --lock-tables, --single-transaction`)

Beispiele:

- Sichern aller Datenbanken mit mysqldump

```
mysqldump -u [Benutzername] -p[Password] -h [Datenbankserver]
       --verbose --all-databases > [Datei]
```

- nur die Datenbankstruktur aber keine Daten sichern

```
mysqldump -u [Benutzername] -p[Password] -h [Datenbankserver]
       --verbose --no-data [Datenbank] > [Datei]
```

- nur Daten aber keine Struktur (kein `drop` oder `create table`)

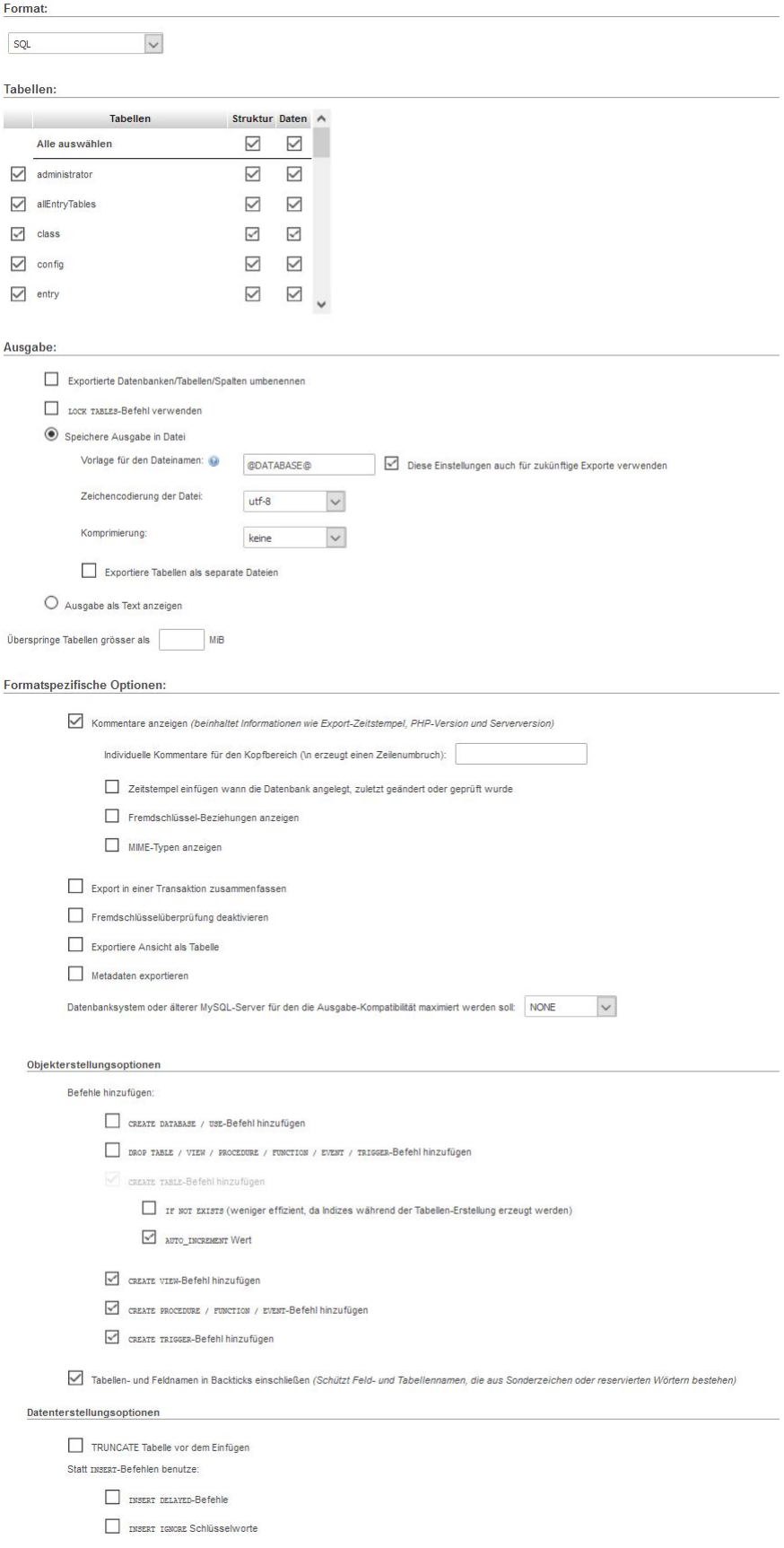
```
mysqldump -u [Benutzername] -p[Password] -h [Datenbankserver] --verbose --no-create-db
       --no-create-info [Datenbank] > [Datei]
```

- Backup in eine existierende Datenbank einspielen

```
mysql -u [Benutzername] -p[Password] [Datenbank] < [Datei]
```

4.3.4 Daten mit phpMyAdmin sichern

Exemplarisch soll hier kurz ein Screenshot gezeigt werden, wie man mit phpMyAdmin Datenbanken sichern kann (Menüpunkt exportieren):


Format:
SQL
Tabellen:
Tabellen
Struktur
Daten
Alle auswählen
administrator
allEntryTables
class
config
entry
Ausgabe:
Exportierte Datenbanken/Tabellen/Spalten umbenennen
LOCK TABLES-Befehl verwenden
Speichere Ausgabe in Datei
Vorlage für den Dateinamen:
@DATABASE@
Diese Einstellungen auch für zukünftige Exporte verwenden
Zeichencodierung der Datei:
utf-8
Komprimierung:
keine
Exportiere Tabellen als separate Dateien
Ausgabe als Text anzeigen
Überspringe Tabellen grösser als
MiB
Formatspezifische Optionen:
Kommentare anzeigen (beinhaltet Informationen wie Export-Zeitstempel, PHP-Version und Serverversion)
Individuelle Kommentare für den Kopfbereich (\n erzeugt einen Zeilenumbruch):
Zeitstempel einfügen wann die Datenbank angelegt, zuletzt geändert oder geprüft wurde
Fremdschlüssel-Beziehungen anzeigen
MIME-Typen anzeigen
Export in einer Transaktion zusammenfassen
Fremdschlüsselüberprüfung deaktivieren
Exportiere Ansicht als Tabelle
Metadaten exportieren
Datenbanksystem oder älterer MySQL-Server für den die Ausgabe-Kompatibilität maximiert werden soll:
NONE
Objekterstellungsoptionen
Befehle hinzufügen:
CREATE DATABASE / USE-Befehl hinzufügen
DROP TABLE / VIEW / PROCEDURE / FUNCTION / EVENT / TRIGGER-Befehl hinzufügen
CREATE TABLE-Befehl hinzufügen
IF NOT EXISTS (weniger effizient, da Indizes während der Tabellen-Erstellung erzeugt werden)
AUTO_INCREMENT Wert
CREATE VIEW-Befehl hinzufügen
CREATE PROCEDURE / FUNCTION / EVENT-Befehl hinzufügen
CREATE TRIGGER-Befehl hinzufügen
Tabellen- und Feldnamen in Backticks einschließen (Schützt Feld- und Tabellennamen, die aus Sonderzeichen oder reservierten Wörtern bestehen)
Datenerstellungsoptionen
TRUNCATE Tabelle vor dem Einfügen
Statt INSERT-Befehlen benutze:
INSERT DELAYED-Befehle
INSERT IGNORE Schlüsselworte

4.3.5 Inkrementelles Backup in MySQL

Für die Point-in-Time Recovery Methode (inkrementelles Backup) von MySQL ist es erforderlich, das Binary Log aktivieren.

In MySQL kann dies in der Konfigurationsdatei (LINUX: my.cnf, XAMPP: mysql/bin/my.ini) eingestellt werden.

Im Bereich [mysqld] muss folgende Zeilen hinzugefügt werden:

```
log_bin = "C:/xampp/mysql/log/mysql-bin.log"
```

Danach ist der Server neu zu starten.

Betrachen Sie den Inhalt des Log-Ordners: `mysql-bin.index, mysql-bin.000001`

Vorgehensweise für Backup:

- Vollbackup erstellen mit mysqldump:

  ```
  mysqldump -u root -p --single-transaction --flush-logs
         --master-data=2 db2backup > full_backup.sql
  ```

- Inkrementelle LOG-Files erstellen mit

  - ```
    mysql> FLUSH LOGS
    ```

    ```
    mysqladmin -u root -p flush-logs
    ```

  - Jedes "Flush Logs" bewirkt, dass ein neues LOG-File im Log-Ordner erstellt wird, welches später gesichert werden und im Bedarfsfall zurückgespielt werden kann

Vorgehensweise für Restore:

- Vollbackup zurückspielen

  ```
  mysql -u root -p db2backup < full_backup.sql
  ```

- Und inkrementelles Backup ergänzen zB

  - ```
    mysqlbinlog.exe c:\xampp\mysql\log\mysql-bin.000002 | mysql -u root -p db2backup
    ```

    bzw.

  - ```
    mysqlbinlog.exe c:\xampp\mysql\log\mysql-bin.000002
           --stop-datetime="2019-01-01 23:34:00" | mysql -u root -p db2backup
    ```

    falls Daten nur bis zu einem bestimmten Zeitpunkt wiederhergestellt werden sollen.

4.3.6 Übungen

- Erstellen Sie in der Tabelle person der Datenbank flugschule 2 neue Einträge.
- Leeren Sie den internen Cachespeicher für die geöffneten Tabellen und sperren Sie die Tabelle person für Schreibzugriffe
- Sichern Sie den Inhalt der Tabelle person mit der SQL-Anweisung `SELECT INTO OUTFILE` in eine delimited ASCII-Datei mit dem Namen `backPerson.txt`. Die einzelnen Daten der Felder sollen in doppelte Anführungszeichen eingeschlossen sein und ein Semikolon soll die einzelnen Daten trennen.
- Heben Sie die Sperre der Tabelle person wieder auf.
    ```

- Erstellen Sie mit dem Dienstprogramm `mysqldump` eine Sicherung der Daten der Tabelle person in die Datei `dumpPerson.sql`.
- Mit welchem Befehl können Sie die Daten wieder einspielen?
- Erstellen Sie mit Hilfe von phpmyadmin ein Backup der Datenbank flugschule, welche Einstellungsmöglichkeiten können Sie hier zusätzlich angeben?
- Testen Sie inkrementelles Backup mit Binary Log:
  - Aktivieren Sie das Binary Log des MySQL-Servers mittels Konfigurationsdatei

    ```
    log_bin = "C:/xampp/mysql/log/mysql-bin.log"
    ```

  - Erstellen Sie folgende Datenbank mit Tabelle und Tupeln:

    ```
    CREATE DATABASE db2backup;

    USE db2backup

    CREATE TABLE tab1 (
           id INT NOT NULL AUTO_INCREMENT,
           info VARCHAR(30) NOT NULL,
           time_created TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
           PRIMARY KEY (id) ) ENGINE=InnoDB;

    INSERT INTO tab1 (info) VALUES ('wert1');
    INSERT INTO tab1 (info) VALUES ('wert2');
    INSERT INTO tab1 (info) VALUES ('wert3');
    ```

  - Erstellen Sie ein vollständiges Backup:

    ```
    mysqldump -u root -p --single-transaction --flush-logs
           --master-data=2 db2backup > full_backup.sql
    ```

    Beobachten Sie den Inhalt des LOG-Ordners (Durch --flush-logs wird eine neue Log-Datei nach dem Vollbackup gestartet).
  - Ergänzen Sie 2 Datensätze:

    ```
    INSERT INTO tab1 (info) VALUES ('wert4');
    INSERT INTO tab1 (info) VALUES ('wert5');
    ```

    Warten Sie etwas und ergänzen Sie 2 weitere Datensätze

    ```
    INSERT INTO tab1 (info) VALUES ('wert6');
    INSERT INTO tab1 (info) VALUES ('wert7');
    ```

  - Kontrollieren Sie den Inhalt von tab1

    ```
    SELECT * FROM tab1;
    ```

  - Erstellen Sie ein inkrementelles Backup-File:

    ```
    mysqladmin -u root -p flush-logs
    ```

    Beobachten Sie den Inhalt des LOG-Ordners wieder wurde ine neue Log-Datei begonnen (mysql-bin. 000003) Die alte Datei (mysql-bin.000002) könnte nun für ein erforderliches Restore gesichert werden (zusätzlich zum letzten Vollbackup)
  - Die Datenwiederherstellung soll nun getestet werden. Löschen Sie dazu nun die Tabelle tab1 und spielen Sie die Daten wieder ein:
    - Letztes Vollbackup einspielen:

      ```
      mysql -u root -p db2backup <full_backup.sql
      ```

Sehen Sie sich den Inhalt von tab1 an (3 Tupel)

* inkrementelles Backup ergänzen:

```
mysqlbinlog.exe c:\xampp\myslq\log\mysql-bin.000002 |
       mysql -u root -p db2backup
```

bzw.

```
mysqlbinlog.exe c:\xampp\myslq\log\mysql-bin.000002
       --stop-datetime="2019-01-01 23:34:00" | mysql -u root -p db2backup
```

- Mit dem Tool `mysqldump` haben Sie die Möglichkeit, Daten aus einer Datenbank in eine andere Datenbank auf einem anderen MySQL-Server zu übertragen.

```
mysqldump [-u<Benutzername> -p<Benutzerkennwort> -h<HostnameDesMySQLServers>
       --opt Datenbankname |mysql [-u<Benutzername> -p<Benutzerkennwort>]
       -hHostnameDesAnderenServers -C Datenbankname
```

- Richten Sie eine 2. xampp-Installation ein, ändern Sie den Port auf 3307 und sichern Sie alle Datenbanken des ersten Servers mithilfe eines physischen Backups (Kopieren aller Datenbanken mit Hilfe des Betriebssystems)