MySQL-Befehle

1 Arbeiten mit Datenbanken

1.1 Datenbank anlegen

Eine Datenbank kann man wie folgt erstellen.

```
CREATE DATABASE db_name;
```

1.2 Existierende Datenbanken anzeigen

Mit dem folgenden Befehl werden alle existierenden Datenbanken angezeigt.

```
SHOW DATABASES;
```

1.3 Datenbank löschen

Um eine Datenbank zu löschen, benötigen Sie den Befehl DROP DATABASE.

```
DROP DATABASE db_name;
```

Um vor dem Löschen sicher zu stellen dass die Datenbank existiert, sollten Sie folgenden Befehl verwenden.

```
DROP DATABASE IF EXISTS db_name;
```

2 Arbeiten mit Tabellen

2.1 Tabelle anlegen

Eine Tabelle kann man mit dem Befehl CREATE TABLE anlegen. Zusätzlich zu diesem Befehl müssen der Tabellenname und die benötigten Spalten mit den jeweiligen Datentypen angegeben werden.

```
CREATE TABLE tbl_name (
id int NOT NULL,
spalte1 varchar(50)
);
```

2.2 Existierende Tabellen anzeigen

Mit dem folgenden Befehl werden alle existierenden Tabellen angezeigt.

```
SHOW TABLES FROM db_name;
```

2.3 Tabelle löschen

Um eine Tabelle zu löschen, benötigen Sie den Befehl DROP TABLE.

```
DROP TABLE tbl_name;
```

Wie bei dem Befehl DROP DATABASE gibt es auch hier die Option IF EXISTS.

```
DROP TABLE IF EXISTS tbl_name;
```

2.4 Spalte hinzufügen

Wenn nachträglich noch eine Spalte in eine Tabelle eingefügt werden soll, benötigen Sie den Befehl ALTER TABLE mit der Option ADD COLUMN.

```
ALTER TABLE tbl_name ADD COLUMN (spalte2 char(50));
```

2.5 Datentyp einer Spalte ändern

Um nachträglich den Datentyp einer Spalte zu ändern, verwendet man ebenfalls den Befehl ALTER TABLE, allerdings jetzt mit der Option MODIFY.

```
ALTER TABLE tbl_name MODIFY (spalte2 varchar(50));
```

2.6 Spalte löschen

Eine Spalte können Sie mit dem Befehl ALTER TABLE und der Option DROP COLUMN löschen.

```
ALTER TABLE tbl_name DROP COLUMN spalte2;
```

3 Umgang mit Datensätzen

3.1 Daten einfügen

Mit dem Befehl INSERT werden Daten in eine vorhandene Tabelle gespeichert.

```
INSERT INTO tbl_name (spalte1, spalte2) VALUES (wert1, wert2);
```

Wollen Sie in eine Spalte keinen Wert eintragen, so lassen Sie diese Spalte einfach aus der Anweisung heraus.

3.2 Daten modifizieren

Um einen Datensatz zu ändern, verwendet man den Befehl UPDATE.

```
UPDATE tbl_name SET spalte1 = neuer_wert1 WHERE id = 1;
```

Mit diesem Befehl wird der vorhandene Wert vom ersten Eintrag (id = 1) mit dem neuen Wert (neuer_wert1) überschrieben.

3.3 Daten löschen

Ein Datensatz wird mit dem Befehl DELETE gelöscht.

```
DELETE FROM tbl_name WHERE id = 1;
```

Wollen Sie die komplette Tabelle leeren, müssen Sie nur die Bedingung entfernen.

```
DELETE FROM tbl_name;
```

4 Abfragen erstellen

4.1 Einfache Abfrage

Die folgende Befehlszeile gibt den gesamten Inhalt einer Tabelle aus.

```
SELECT * FROM tbl_name;
```

Sollen nur einzelne Spalten von der ganzen Tabelle angezeigt werden, müssen Sie den Operator * durch die Spaltennamen ersetzen

```
SELECT spalte1, spalte2 FROM tbl_name;
```

4.2 Einfache Abfrage mit DISTINCT

Mit der Option DISTINCT können Duplikate ausgefiltert werden.

```
SELECT DISTINCT spalte1 FROM tbl_name;
```

4.3 Abfrage mit Bedingung

Der folgende Befehl gibt die komplette Zeile aus, bei der in der Spalte id eine 1 steht.

```
SELECT * FROM tbl_name WHERE id = 1;
```

Wenn Sie nach einem Wort suchen, müssen Sie dieses in zwei Hochkommas setzen.

```
SELECT * FROM tbl_name WHERE spalte1 = 'wort';
```

Um die Datensätze zwischen 10 und 20 auszugeben können Sie BETWEEN verwenden.

```
SELECT * FROM tbl_name WHERE id BETWEEN 10 AND 20;
```

4.4 Abfrage mit Platzhalter

Wenn Sie nicht genau wissen, an welcher Stelle das Wort steht welches Sie suchen, können Sie den Platzhalter % verwenden. Der Platzhalter % steht für beliebig viele Zeichen.

Weiterhin müssen Sie das = durch LIKE ersetzen.

```
SELECT * FROM tbl_name WHERE spalte1 LIKE '%wort%';
```

Ein weiterer Platzhalter ist der Unterstrich. Dieser steht für genau ein Zeichen.

```
SELECT * FROM tbl_name WHERE spalte1 LIKE 'w__t';
```

4.5 Verknüpfung von Bedingungen

Werden zwei Bedingungen mit einem AND verknüpft, müssen beide Bedingungen erfüllt werden.

```
SELECT * FROM tbl_name WHERE id > 10 AND id < 20;
```

Werden zwei Bedingungen mit einem OR verknüpft, muss mindestens eine Bedingung erfüllt sein.

```
SELECT * FROM tbl_name WHERE id = 10 OR id = 20;
```

Mit der Bedingung NOT kann man das Ergebnis einer Bedingung negieren.

```
SELECT * FROM tbl_name WHERE id < 10 AND NOT id = 5;
```

Werden mehrere Bedingungen miteinander verknüpft, müssen gegebenenfalls Teile der Abfrage in Klammern gesetzt werden.

```
SELECT * FROM tbl_name WHERE id < 10 OR (spalte1 = 'wort' AND id = 5);
```

5 6. Aggregatfunktionen nutzen

5.1 Anzahl der Datensätze ermitteln

Um die Anzahl der Datensätze zu ermitteln auf die eine Abfrage zutrifft, gibt es die Funktion COUNT.

```
SELECT COUNT(*) FROM tbl_name WHERE spalte1 = 'wort';
```

5.2 Summierung

Mit dem Befehl SUM kann man die Summe der Werte einer Spalte (hier preis) ermitteln.

```
SELECT SUM(preis) FROM tbl_name;
```

5.3 Durchschnitt

Möchte man den Durchschnitt aller Werte einer Spalte erhalten, benutzt man die Funktion AVG.

```
SELECT AVG(preis) FROM tbl_name;
```

5.4 Maximalwert

Die Funktion MAX ermittelt den maximalen Spaltenwert.

```
SELECT MAX(preis) FROM tbl_name;
```

5.5 Minimalwert

Das Gegenteil zu der Funktion MAX ist die Funktion MIN.

```
SELECT MIN(preis) FROM tbl_name;
```

5.6 Gruppierung

Mit der Funktion GROUP BY können gleiche Ergebnisse gruppiert werden. Mit Hilfe der folgenden Befehlskombination kann man somit die Anzahl der doppelten Einträge in der Spalte „spalte1" ermitteln.

```
SELECT spalte1, COUNT(*) FROM tbl_name GROUP BY spalte1;
```

5.7 HAVING

Die Funktion HAVING ermöglicht das überprüfen von Bedingungen bei aggregierten Werten. Im folgenden Beispiel werden doppelte Einträge in der Spalte „spalte1" gezählt und nur die angezeigt, die mehr als fünf mal gefunden wurden.

```
SELECT spalte1, COUNT(*) FROM tbl_name GROUP BY spalte1 HAVING COUNT(*) > 5
ORDER BY spalte1;
```

5.8 Sortierung

Die Ergebnisse einer Abfrage können mit der Funktion ORDER BY sortiert werden.

```
SELECT * FROM tbl_name ORDER BY spalte1, spalte2;
```

Um die Sortierreihenfolge fest zu legen gibt es die Schlüsselworte ASC (ascending = aufsteigend) und DESC (descending = absteigend).

```
SELECT * FROM tbl_name ORDER BY spalte1 ASC;
SELECT * FROM tbl_name ORDER BY spalte1 DESC;
```

Wenn kein Schlüsselwort verwendet wird, wird das Ergebnis aufsteigend sortiert.

6 Weiterführendes

MySQL Homepage:
http://www.mysql.de

MySQL Dokumentation:
http://dev.mysql.com/doc/

phpMyAdmin - PHP-Administration-Oberfläche für MySQL
http://www.phpmyadmin.net