1 Data Manipulation Language DML

Neben Daten auslesen, muss man sie auch verändern können. Hierzu dient ein Teilbereich von SQL, die Data Manipulation Language. Es zählen folgende Aktionen hierzu:

- Daten einfügen
- Daten ändern
- Daten löschen

Diese Aktionen können teilweise durch mehrere Statements erfolgen.

2 Daten einfügen

Hierzu zählen:

- `INSERT`
- `REPLACE`
- `LOAD DATA INFILE`

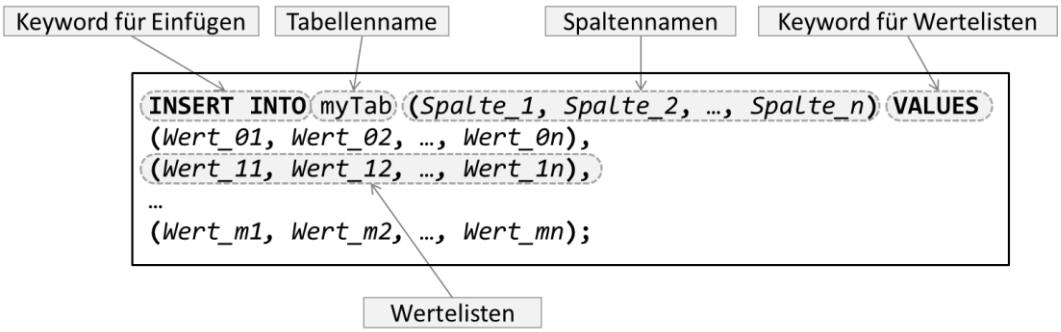
Letzteres stellt einen Sonderfall dar und ermöglicht es, CSV formatierte Daten in die Datenbank zu laden. Der große Vorteil ist die hohe Performance, jedoch handelt es sich hier um einen sehr speziellen Befehl, welcher bei unterschiedlichen RDBMS eigene Syntaxregeln gehorcht. Insofern wird an dieser Stelle lediglich auf `INSERT` und `REPLACE` eingegangen.

Für die folgenden Beispielstatements gehen wir von folgender Tabellenstruktur aus:

Person
P_ID
Vorname
Nachname
GebDat

2.1 Das INSERT Statement

Dies ist das Standardstatement, um Daten in eine Tabelle zu schreiben: folgender Syntax gilt für das Einfügen von mehreren Datensätzen in einem Statement:



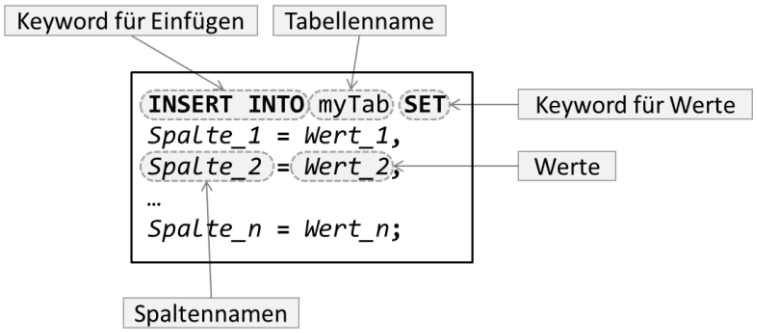
Die Angabe der Spaltennamen ist optional – die DB geht ohne Spaltenangaben von der Anzahl und Reihenfolge der Tabellenstruktur aus dem `CREATE TABLE` Statement aus. Um Fehler zu vermeiden, wird jedoch auf jeden Fall geraten, die Spalten immer explizit anzugeben.

Die Spaltenanzahl und Spaltenreihenfolge müssen denen der Werte entsprechen. Werte, welche keine Zahlen sind, müssen in Hochkommas gesetzt werden.

Beispiel:

```
INSERT INTO Person (P_ID, Vorname, Nachname, GebDat) VALUES
(1, "Peter", "Schmid", "2000-03-12"),
(2, "Michaela", "Huber", "1998-07-08");
```

Für den Fall, dass lediglich ein Datensatz geschrieben werden muss, gibt es eine zweite Syntaxvariante:



Für Einzeldatensätze ist diese Variante vorzuziehen, da sie weniger Fehleranfällig ist. Die Werte stehen hier immer neben den Spalten.

Beispiel:

```
INSERT INTO Person SET
P_ID = 3,
Vorname = "Hans",
Nachname = "Maier",
GebDat = "2001-11-01";
```

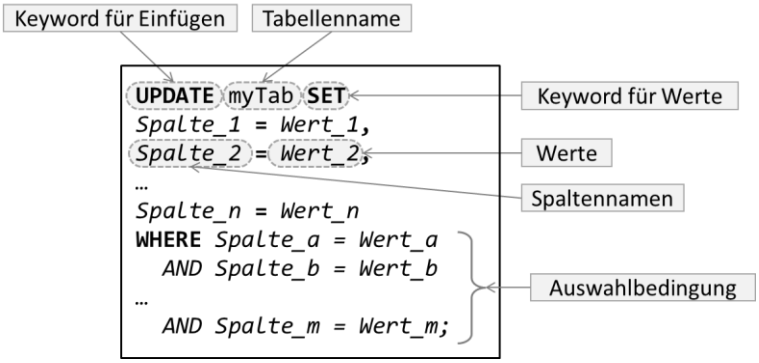
Spalten, welche im `INSERT` Statement nicht berücksichtigt werden, erhalten entweder `NULL`, oder den Defaultwert, sofern er beim `CREATE TABLE` Statement angegeben wurde. Wenn ein Datensatz mit der gleichen P_ID (als Primärschlüssel) bereits existiert, wird eine Fehlermeldung ausgegeben und der Datensatz wird nicht geschrieben.

2.2 Das REPLACE Statement

Das `REPLACE` Statement ist identisch mit dem `INSERT` Statement, es wird lediglich anstatt `INSERT` eben `REPLACE` geschrieben – das gilt für beide `INSERT` Varianten. Prinzipiell bewirken sie das Gleiche – der Unterschied liegt darin, dass das `REPLACE` Statement für den Fall, dass der Primärschlüssel des zu schreibenden Datensatzes bereits vorhanden ist, der alte Datensatz vorab gelöscht wird. Er wird also „ersetzt“.

3 Daten ändern

Wenn ein vorhandener Datensatz angepasst werden muss, wird das „`UPDATE`“ Statement verwendet. Es ist vom Aufbau her eine Mischung aus `INSERT` und dem `SELECT` Statement:



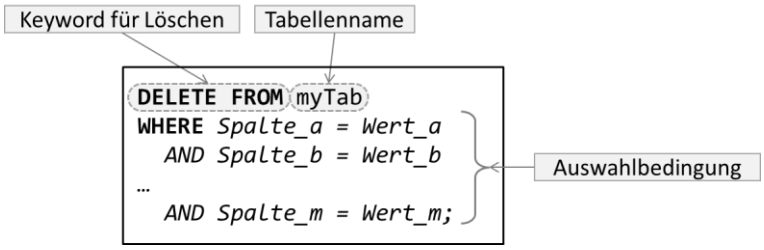
Die `WHERE` Bedingung bestimmt, für welchen Datensatz die Anpassung gemacht wird. Wie beim `SELECT` Statement auch, führt das Weglassen der `WHERE` Bedingung dazu, dass alle Datensätze ausgewählt werden – sprich alle Datensätze würden geändert werden! Also Vorsicht!

Beispiel:

```
UPDATE Person SET
Vorname = "Hannes",
Nachname = "Mayer"
WHERE P_ID = 3;
```

4 Daten löschen

Daten werden mit „DELETE“ gelöscht. Auch hier wird mit Hilfe einer WHERE Bedingung ausgewählt, welche Daten zu löschen sind:



Wie beim UPDATE Statement auch gilt, dass ohne eine WHERE Bedingung alle Datensätze verarbeitet, sprich gelöscht werden!

5 Aufgabenstellung

Führen Sie nun alle folgenden Aufgaben durch und speichern die Statements in einem File namens „TaskDDL.sql“. Sie können hierfür mit CREATE DATABASE dmlDB; und USE dmlDB; eine neue Datenbank nutzen. Prüfen Sie nach jeder DML Aktion das Ergebnis mit einem SELECT Statement. Erstellen sie zuerst mit folgendem Statement eine Tabelle für Produktdaten:

```
CREATE TABLE Produkt (
  PID INT PRIMARY KEY,
  Name VARCHAR(64),
  Hersteller VARCHAR(64),
  Preis DECIMAL(8,2),
  EinfDatum DATE
);
```

Fügen sie nun folgende Daten in einem Statement ein:

PID:	Name:	Hersteller:	Preis:	Einführungs-Datum:
1	Schraubendreherset	Hazet	29,99	01.01.2019
2	Hammer	Mannesmann	12,99	04.10.2019
3	Seitenschneider	Knipex	10,99	08.11.2019
4	Rohrzange	Knipex	12,99	09.11.2019

Ergänzen sie nun folgenden Datensatz:

PID:	Name:	Hersteller:	Preis:	Einführungs-Datum:
5	Cuttermesser	Stanley	8,99	03.02.2020

Ergänzen Sie nun folgenden Datensatz:

PID:	Name:	Hersteller:	Preis:	Einführungs-Datum:
5	Cuttermesser	Stanley	9,99	03.02.2020

Überlegen Sie, warum ein Fehler aufgetreten ist:

Tauschen sie nun `INSERT` durch `REPLACE` aus und führen das Statement nochmal aus. Warum funktioniert es nun?

Ändern Sie nun den Wert des Preises wieder zurück auf 8,99. Nutzen Sie hierbei das `UPDATE` Statement. Ändern Sie anschließend das Einführungsdatum aller Knipex Werkzeuge auf den 10.11.2019. Nutzen sie hierfür nur ein einziges Statement.

Zum Schluss löschen sie alle Werkzeuge, welche mindestens 12,99 kosten.

6 Zusatzaufgabe

Informieren sie sich über den LOAD DATA INFILE Befehl und erzeugen ein Textfile mit passenden Daten, welches mit diesem Befehl in die Tabelle geladen werden kann. Die Informationen für den LOAD DATA INFILE Befehl finden Sie im Klassenlaufwerk in der Datei SQL_Syntax.pdf.