SQL Syntax-Referenz mit Beispielen

Eine stark reduzierte Übersicht anhand von Beispielen

Tabellen erstellen: CREATE TABLE

```
CREATE TABLE  movies (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  title VARCHAR(64),
  year INTEGER,
  mppa VARCHAR(5),
  director VARCHAR(5),
  runtime INTEGER
);
```

Erzeugt eine einfache Tabelle mit 6 Feldern (Spalten), eins davon ist der Primärschlüssel, der automatisch vergeben wird.

```
CREATE TABLE moviecast (
  movieid INTEGER NOT NULL,
  actor VARCHAR(64),
  impersonated VARCHAR(64),
  FOREIGN KEY movieid REFERENCES movies(id)
);
```

Erzeugt eine Tabelle mit einem Fremdschlüssel movieid, der auf das id*-Feld der* movies*-Tabelle verweist.*

Neben INTEGER und VARCHAR gibt es auch noch weitere Datentypen, z.B. DATETIME für Datum und Uhrzeit, TEXT für längeren Text und BLOB für nicht genauer spezifizierte Daten, z.B. Bild- oder Tondaten. Die verfügbaren Datentypen hängen vom verwendeten DBMS ab.

Neue Datensätze hinzufügen: INSERT

```
INSERT INTO movies(id, title, year, mppa, director, runtime) VALUES (1,“Ronin“, 1998, “R“, “John Frankenheimer“, 122)
```

Die id könnte weggelassen werden, weil sie automatisch eingefügt wird (AUTOINCREMENT).

```
INSERT INTO moviecast (movieid, actor, impersonated) VALUES ('1,“Jean Reno“, “Vincent“)
```

Bestehende Datensätze anpassen: UPDATE

```
UPDATE movies SET  mppa = “PG-13“ WHERE id = 1
```

Ändert die Altersfreigabe für den Film „Ronin“. Die WHERE-Klausel verwendet den Primary Key, um den Film zu identifizieren.

```
UPDATE movies SET  mppa = “PG-13“ WHERE id = 1 or director = “Tim McCanlies“
```

Ändert die Altersfreigabe für den Film mit id 1 und alle Filme des Regisseurs Tim McCanlies.

```
UPDATE moviecast SET  actor=“Robert De Niro“, impersonated=“Sam“ WHERE actor=“Jean Reno“
```

Gefährlich, weil die WHERE-Klausel eventuell auf mehrere Datensätze zutrifft. Mit diesem Update werden sie alle modifiziert!

Datensätze löschen: DELETE

```
DELETE FROM movies WHERE runtime < 100 and director=“Luc Besson“
```

Löscht alle Datensätze für Filme aus der movies-Tabelle, welche kürzer als 100 Minuten sind und Luc Besson als Regisseur haben.

Tabellen löschen: DROP TABLE

```
DROP TABLE movies
```

```
DROP TABLE moviecast
```

Achtung: Keinerlei Warnung! Die Tabelle inklusive aller Datensätze in ihr wird gelöscht.

Einfache Datenbankabfragen: SELECT

SELECT * FROM movies

Liefert sämtliche Datensätze der movies-*Tabelle.*

SELECT * FROM movies WHERE runtime > 100

Liefert die Datensätze aller Filme, die länger als 100 Minuten dauern

SELECT title, runtime FROM movies WHERE year = 1998 AND director=“Tony Kaye“

Liefert Titel und Spieldauer aller Filme, die 1998 von Regisseur Tony Kaye produziert wurden

Wiederholungen vermeiden

SELECT name from genre

SELECT distinct name from genre

Aggregationsfunktionen: Zählen und Summieren

SELECT count(name) FROM genre

SELECT count(distinct name) FROM genre

SELECT sum(runtime) FROM movies WHERE year = 1998

Nach Aggregat gruppieren

SELECT director, count(id) FROM movies GROUP BY director

Gruppiert alle Filmdatensätze nach dem Regisseur und zählt dann nicht mehr sämtliche Datensätze, sondern für jede Gruppe (d.h. Jeden Regisseur), wieviele Datensätze in dieser Gruppe sind. Liefert also im obigen Fall die Antwort darauf, wieviele Filme jeder Regisseur gemacht hat.

Nur bestimmte Gruppierungen auswählen

SELECT director, count(id) FROM movies GROUP BY director HAVING count(id) > 1

Tabellen verknüpfen

SELECT title FROM movies,genre WHERE id = movieid AND name = “Comedy“

Liefert den Titel aller Komödien. Die Klausel id = movieid ist nötig, damit aus beiden Tabellen nur zusammengehörende Kombinationen ausgewählt werden.

SELECT title FROM movies,genre,moviecast WHERe id=genre.movieid AND id=moviecast.movieid AND name=“Sci-Fi“ and actor=“Bruce Willis“

Wenn Attribute gleich heissen, muss die Tabellenbezeichnung ebenfalls angegeben werden: id = genre.movieid