

Datenbank-Operationen

Neue Datenbank anlegen: `CREATE DATABASE Datenbankname;`

Beispiel: `CREATE DATABASE testdb;`

Datenbank löschen: `DROP DATABASE Datenbankname;`

Datenbank kopieren: `BACKUP DATABASE Datenbankname TO DISK = ,Speicherort‘;`

Beispiel: `BACKUP DATABASE testdb TO DISK =
,E:\backup\DB\testdb.bak‘;`

Tabellen-Operationen

Neue Tabelle anlegen: `CREATE TABLE Tabellename (
 Spalte1 Datentyp,
 Spalte2 Datentyp,
 ...
 Spalten Datentyp
);`

Beispiel: `CREATE TABLE kunde (
 K_ID int,
 Nachname varchar(60),
 Vorname varchar(40),
 ...
 GebDatum Datum
);`

Eine Tabelle löschen: `DROP TABLE Tabellename;`

Alle Datensätze in Tabelle löschen: `TRUNCATE TABLE Tabellename;`

Neue Spalte anlegen: `ALTER TABLE Tabellename ADD Spaltenname Datentyp;`

Beispiel: `ALTER TABLE kunde ADD Adresse varchar(30);`

Spalte löschen: `ALTER TABLE Tabellename DROP COLUMN Spaltenname;`

Beispiel: `ALTER TABLE kunde DROP COLUMN Zahlung;`

Spalte umbenennen: `ALTER TABLE Tabellename RENAME COLUMN AlterName
NeuerName;`

Beispiel: `ALTER TABLE kunde RENAME COLUMN Mail Email;`

Datentyp einer Spalte ändern:

`ALTER TABLE Tabellename MODIFY COLUMN Spaltenname Datentyp;`

Beispiel: `ALTER TABLE kunde MODIFY COLUMN Telefon varchar(20);`

Einschränkungen bei Tabellen:

NOT NULL: In der Spalte sind leere Felder NICHT zugelassen.

UNIQUE: Stellt sicher, dass alle Werte in der Spalte verschieden sind.

PRIMARY KEY: Das ist Kombination aus den beiden obigen Einschränkungen.

FOREIGN KEY: Verhindert Aktionen, die Verknüpfungen zwischen Tabellen zerstören.

CHECK: Stellt sicher, dass die Werte in der Spalte bestimmte Bedingung(en) erfüllen.

DEFAULT: Legt den Standardwert für die Spalte fest, falls kein Wert angegeben ist.

CREATE INDEX: Wird zum schnellen Abrufen von Daten benutzt.

Beispiel für NOT NULL:

```
CREATE TABLE person (  
    ID NOT NULL,  
    Nachname varchar(40) NOT NULL,
```

GebDatum Date);

Tabelle erzeugen mit einer Spalte mit Einschränkung UNIQUE (MySQL-Version):

```
CREATE TABLE gamer (  
    ID int,  
    Game_ID int,  
    Nickname varchar(40) NOT NULL,  
    Level int,  
    PRIMARY KEY (ID),  
    UNIQUE (Game_ID));
```

Primary Key aus zwei Spalten bilden (MySQL-Version):

```
CREATE TABLE gamer (  
    ID int,  
    Game_ID int,  
    Nickname varchar(40) NOT NULL,  
    Level int,  
    CONSTRAINT PK_Gamer PRIMARY KEY (ID, Game_ID));
```

Primary Key nachträglich definieren:

```
ALTER TABLE Person ADD PRIMARY KEY (ID);
```

```
ALTER TABLE Person ADD CONSTRAINT PK_Person PRIMARY KEY (ID, GebDatum);
```

Löschen eines Primary Keys:

```
ALTER TABLE Person DROP PRIMARY KEY;
```

FOREIGN KEY definieren:

Situation: Zwei Tabellen: Kunde, Bestellung

Beispiel FOREIGN KEY in Tabelle Bestellung:

```
CREATE TABLE Bestellung (  
    OrderID int,  
    OrderNr int NOT NULL,  
    KundeID int,  
    PRIMARY KEY (OrderID),  
    FOREIGN KEY (KundeID) REFERENCES Kunde(KundeID));
```

Foreign Key löschen:

```
ALTER TABLE Bestellung DROP FOREIGN KEY KundeID;
```

Auswahl-Abfragen:

Beginnen grundsätzlich mit dem SQL Befehl SELECT

Syntax: SELECT Was FROM Woher

Beispiel: SELECT Nachname, Vorname FROM kunde;

(Wählt die Spalten Nachname und Vorname aus der Tabelle kunde aus)

 SELECT * FROM kunde;

(Wählt die kompletten Datensätze der Tabelle kunde aus)

 SELECT * FROM kunde WHERE Kunde_ID = 123;

(Wählt den kompletten Datensatz aus, dessen ID = 123 ist)

 SELECT Nachname, Vorname, PLZ FROM kunde WHERE Kunde_ID > 156 AND
Kunde_ID < 176;

(Wählt die Nachnamen, die Vornamen und zugehörigen PLZ aus der Tabelle kunde aus, mit der Bedingung, dass die Kunde_ID zwischen 157 und 175 liegt)

Das folgende Beispiel

 SELECT Nachname, Vorname FROM kunde;

Kann auch so geschrieben werden:

 SELECT kunde.Nachname, kunde.Vorname FROM kunde;

Oder alternativ:

 SELECT k.Nachname, k.Vorname FROM kunde k;

Abfrage bei Strings oder Character:

 SELECT Vorname, PLZ FROM kunde WHERE Nachname = ‚Müller‘;

 SELECT Vorname, PLZ FROM kunde WHERE Nachname <> ‚Müller‘;

Auswahl über zwei Tabellen:

Student

ID	Name	Semester	Kurs	KursID
1	Snoopy	3	Taktik	11
2	Schroeder	6	Musik	16
3	Linus	5	Politik	13
4	Woodstock	8	Flugunterricht	23
5	Lucy	4	Medizin	30

Kurs

KursID	Bezeichnung
11	Taktik
12	Mathematik
13	Politik
14	Literatur
15	Sprachen
16	Musik
17	Chemie
18	Kunst

1. INNER JOIN

```
SELECT Student.Name, Student.Semester, Kurs.Bezeichnung
```

```
FROM Student
```

```
INNER JOIN Kurs
```

```
ON Student.KursID = Kurs.KursID
```

Resultat: Liefert den Studentennamen, die Semesteranzahl (aus der ersten Tabelle) und die Kursbezeichnung aus der zweiten zurück, die eine Entsprechung in beiden Tabellen haben.

2. LEFT JOIN

```
SELECT Student.Name, Student.Semester, Kurs.Bezeichnung
```

```
FROM Student
```

```
LEFT JOIN Kurs
```

```
ON Student.KursID = Kurs.KursID
```

Liefert die Übereinstimmungen bei der KursID, die in beiden Tabellen enthalten ist. Bei den Einträgen, wo es keine Übereinstimmung gibt wird die Kursbezeichnung mit NULL ausgegeben.

3. RIGHT JOIN

```
SELECT Student.Name, Student.Semester, Kurs.Bezeichnung  
FROM Student  
RIGHT JOIN Kurs  
ON Student.KursID = Kurs.KursID
```

Liefert die Übereinstimmungen bei der KursID, die in beiden Tabellen enthalten ist. Bei den Einträgen, wo es keine Übereinstimmung gibt werden die Namen und Semester mit NULL ausgegeben.

4. FULL JOIN

```
SELECT Student.Name, Student.Semester, Kurs.Bezeichnung  
FROM Student  
RIGHT JOIN Kurs  
ON Student.KursID = Kurs.KursID
```

Liefert alle Einträge aus beiden Tabellen und für diejenigen Einträge, wo es in der anderen Tabelle keine Entsprechung gibt wird in den Spalten NULL ausgegeben.