SQL-Cheatsheet

**Anlegen von Tabellen:** **Anlegen einer Beziehungstabelle:**

**create** **table** fach(

id **int primary key not null**,

name **varchar**(*20*)

);

**create** **table** lehrkraft(

id **int primary key not null**,

rufname **varchar**(*80*),

familienname **varchar**(*100*)

);

**create** **table** unterrichtet(

lehrkraft\_id **int** **references** lehrkraft(id),

fach\_id **int** **references** fach(id),

**primary key** (lehrkraft\_id, fach\_id)

);

**create** **table** benutzer (

id **int primary key auto\_increment**,

rufname **varchar**(*80*) **not null**,

familienname **varchar**(*120*) **not null**,

geburtsdatum **date**,

telefonnummer **varchar**(*30*)

);

**create** **table** buch (

id **int primary key auto\_increment**,

isbn **varchar**(*17*),

titel **varchar**(*80*),

autor **varchar**(*100*),

benutzer\_id **int** **references** benutzer(id)

);

**delete from** benutzer

**where** id **=** *2*;

**Löschen von Datensätzen:**

In Klammern stehen jeweils die Spaltenwerte eines Datensatzes. Es können beliebig viele Datensätze hintereinander angegeben werden.

**update** benutzer

**set**

plz **=** '86529', strasse **=** 'Geisendelder Straße', nummer **=** '1a', ort **=** 'Manching'

**where** id **=** *2*;

**Ändern von Datensätzen:**

Angabe der Spalten, in die Werte eingefügt werden sollen. Um die Werte für die Spalte id kümmert sich die Datenbank (**auto\_increment**, s.o.!).

**insert into** benutzer (rufname, familienname, geburtsdatum, telefonnummer)

**values**

( 'Beate', 'Beispiel', '1974-09-30', '12345'),

( 'Benno', 'Beispiel', '1990-07-02', '67890');

**insert into** buch (isbn, titel, autor, benutzer\_id)

**values**

( '978-3-7891-1851-7', 'Pippi Langstrumpf', 'Astrid Lindgren', *2*);

**Einfügen von Datensätzen:**

Abfragen

Die Spalte id ist **Primärschlüssel** der Tabelle. Das Schlüsselwort **auto\_increment** weist die Datenbank an, die Werte für die Spalte id beim Einfügen eines neuen Datensatzes automatisch zu füllen.

Die Spalte benutzer\_id beinhaltet **Fremdschlüssel**, die auf die entsprechenden Werte der Spalte id in der Tabelle benutzer verweisen. Mit dem **references**-Schlüsselwort teilen wir dies der Datenbank mit.

Die Spalten rufname und familienname sind **Pflichtfelder** der Tabelle, d.h. die Datenbank soll eine Fehlermeldung ausgeben, wenn der Benutzer versucht, **null** als Wert einzufügen. Das wird durch Angabe von **not null** bewirkt.

Die Spalten lehrkraft\_id und fach\_id bilden **zusammen** den **Primärschlüssel** der Tabelle (**kombinierter Primärschlüssel**).

Die Tabelle lehrkraft\_fach speichert die n:m-Relation zwischen lehrkraft und fach. So eine Tabelle nennt man Beziehungstabelle.

**alter** **table** fach

**add** istFremdsprache **boolean**;

**Spalte einfügen:**

**alter** **table** fach **drop** name;

**Spalte löschen:**

**drop** **table** fach;

**Tabelle löschen:**

**varchar(20)** Zeichenkette mit Maximallänge (hier: 20)

**text** Zeichenkette mit Maximallänge 65535 ( = 216 – 1)

**longtext** Zeichenkette mit Maximallänge 4 294 967 295 ( = 232 – 1)

**boolean** Wahrheitswert (true oder false)

**decimal(8,2)** Dezimalzahl fester „Länge“ (hier: 8 Ziffern, davon 2 Nachkommastellen)

**float** Fließkommazahl mit ca. 7 Dezimalstellen

**double** Fließkommazahl mit ca. 15 Dezimalstellen

**date** Datum

**time** Uhrzeit

**datetime** Datum und Uhrzeit (auch: timestamp)

**Datentypen:**

Ein \* statt der Spaltenliste bedeutet: Die Ergebnistabelle soll **alle** Spalten der Tabelle Benutzer enthalten.

**select** rufname, familienname

**from** benutzer

**where**

ort **=** 'Manching'

**select** **\***

**from** benutzer

**where**

ort **=** 'Manching' **and** familienname **like** '\_p%'

**select** rufname, familienname, titel

**from** benutzer, buch

**where**

*buch*.*benutzer\_id* **=** *benutzer*.*id*

**select** rufname, familienname, name **as** 'Fach'

**from** lehrkraft, fach, lehrkraft\_fach

**where**

*lehrkraft*.*id* **=** *lehrkraft\_fach*.*lehrkraft\_id* **and**

*fach*.*id* **=** *lehrkraft\_fach*.*fach\_id*

Vergleicht man Zeichenketten mit like statt mit Gleichheitszeichen (=), so kann nach ähnlichen Werten gesucht werden. Dabei haben die Platzhalter % und \_ eine besondere Bedeutung: % steht für beliebig viele (auch: 0) beliebige Zeichen, \_ steht für genau ein beliebiges Zeichen.

familienname **like** '\_p%' ist **true** bei allen Datensätzen, bei denen das Attribut familienname als zweiten Buchstaben ein p besitzt.

Die Ergebnistabelle soll die Spalten rufname und familienname besitzen (**Projektion**).

Die Abfrage bezieht sich auf die Tabelle benutzer.

Die **where**-Bedingung schränkt ein, welche Zeilen der Tabelle benutzer in die Ergebnistabelle übernommen werden sollen (**Selektion**)

**Join** der Tabellen benutzer und buch: Alle Zeilen der Tabelle benutzer werden mit allen Zeilen der Tabelle buch kombiniert. Um die Ergebnistabelle auf diejenigen Zeilen zu beschränken, die zusammenpassen, erfolgt die **Selektion** mittels *buch*.*benutzer\_id* **=** *benutzer*.*id*

Will man zwei über eine **n:m-Relation** verbundene Tabellen bei einer Abfrage kombinieren, so muss man über die Beziehungstabelle **joinen**.

Hier knicken, mit Kleber bestreichen und ins Heft einkleben!