Einführung in Datenbanksysteme

Tutorium: SQL (DDL & DML)

Tutoren

Mit Folienmaterial aus der Vorlesung und DBPRA

Fachgebiet Datenbanksysteme und Informationsmanagement
Technische Universität Berlin

http://www.dima.tu-berlin.de/

Überblick

- Wiederholung
  - DQL (Data Query Language)
- DDL (Data Description Language)
  - Tabellen erstellen und verändern
- DML (Data Manipulation Language)
  - Einfügen, Ändern und Löschen

Wiederholung

- Grundgerüst: `SELECT … FROM … WHERE …`
  - `SELECT`: Ausgabeformat (Projektion von Attributen in RA)
  - `FROM`: Abgefragte Tabellen
  - `WHERE`: Bedingungen für Ergebnistupel (Selektion in RA)

- Join:
  - Implizite Syntax
    - Tabellen in `FROM-Klausel` listen
    - Join Bedingung in `WHERE-Klausel` einfügen
  - Explizite Syntax
    - `FROM … JOIN … ON …`

Wiederholung

- Sortierung
  - `ORDER BY … <ASC | DESC>`
- Duplikateleminierung
  - `SELECT DISTINCT …`
- Aggregatfunktionen
  - `COUNT(), SUM(), AVG(), MIN(), MAX()`
- Gruppierung
  - `GROUP BY …`
- Auswahl von Gruppen
  - `HAVING …`
- Mengenoperationen
  - `… <UNION|INTERSECT|EXCEPT> …`

SQL-Anweisungen

DQL	DML	DDL	DCL	Transaktions-steuerung
SELECT … FROM … WHERE … …	INSERT … UPDATE … DELETE … …	CREATE DATABASE … CREATE TABLE … ALTER TABLE … …	GRANT SELECT ON *Tablename* TO *User* …	COMMIT ROLLBACK SAVEPOINT …

- DQL (Data Query Language)
- DML (Data Manipulation Language)
- DDL (Data Description Language)
- DCL (Data Control Language)

Beachtet die Unterschiede bei Systemen.
Hier durchgehend MySql

DDL: Databank und Tabellen erstellen

- Datenbank erstellen
  - `CREATE DATABASE dbname` (+ optionale Eigenschaften)

- Tabellen erstellen
  - `CREATE TABLE <IF NOT EXISTS> Tablename (`
    `spaltenname spaltentyp constraints`
    `)`

- Beispiel
  - ```
    CREATE TABLE Person (
      name VARCHAR(30) NOT NULL,
      geburtstag DATE NOT NULL,
      geburtsort VARCHAR(30),
      PRIMARY KEY (name, geburtstag)
    )
    ```
    ```

DDL: Zeichensatz & Sortierung angeben

Beispiel:

```
CREATE DATABASE person CHARACTER SET latin1 COLLATE
latin1_german1_ci;
```

Zeichensatz: latin1 (entspricht ISO-8859-1)

Sortierregeln: latin1_german1_ci

`SHOW COLLATION;` Alle verfügbaren Zeichensätze anzeigen

`SHOW VARIABLES;` die MySQL-Character-Einstellungen ansehen

Sortierung MySql

Bezeichnung	Bemerkung	MySql-Variante	Sortierregeln
Standard	(schwedisch)	default	ä/ö/ü nach z
General	(englisch)	latin1_general_ci latin1_general_cs	ä/ö/ü/ß nach a/o/u/s
German1	DIN 1, Duden, für Wörter	latin1_german1_ci latin1_german1_cs	ä/ö/ü/ß wie a/o/u/s
German2	DIN 2, Telefonbuch, für Namen	latin1_german2_ci latin1_german2_cs	ä/ö/ü/ß wie ae/oe/ue/ss

„..._ci“ bedeutet "Case Insensitive, „..._cs“ bedeutet Case Sensitive

Weiteres

- Collation anzeigen

```
SHOW COLLATION;
```

- Tabellen innerhalb einer Datenbank anzeigen

```
SHOW TABLES;
```

- Detallierte Infos über Tabellen

```
SHOW TABLE status;
```

- CREATE Table Query Anzeigen

```
SHOW CREATE TABLE person;
```

Tabelle über “SELECT” erstellen

- Beispiel:

Relationenschema wird übernommen und Inhalte werden kopiert

```
CREATE TABLE Piraten
AS
   SELECT *
   FROM Teilnehmer
   WHERE Bootsklasse = ‘Pirat’;
```

Beispiel: Nur die Struktur übernehmen (aber keine Datensätze vorhanden)

```
CREATE TABLE Piraten
AS
   SELECT *
   FROM Teilnehmer
   -- Es gibt keine Tupel mit der Bootsklasse XYZ in der Instanz
   WHERE Bootsklasse = ‘XYZ’;
```

Datentypen (Hier MySql)

Spaltentypen	Beschreibung
INT, INTEGER	Integer normaler Größe.
FLOAT[(*M*)]	Kleine Fließkommazahl, *M* ist die Gesamtzahl an Ziffern
DOUBLE[(*M*)]	Fließkommazahl normaler Größe (mit doppelter Genauigkeit), *M* ist die maximale Anzahl an Ziffern
DECIMAL[(*M*)]	Gepackte „exakte“ Festkommazahl. ***M*** ist die Gesamtzahl von Dezimalstellen, Berechnungen haben eine Genauigkeit von 65 Stellen
CHAR(x)	Text-Datentyp mit x Stellen, 0 <= x <= 254, CHAR(0) für NULL und ,‘ Belegt immer Speicherplatz für x Stellen!
VARCHAR(x) String	Strings variabler Länge. Diese kann zwischen 0 und 65.535 liegen Belegt nur benötigten Speicherplatz!
DATE	Datum-Datentyp FORMAT: ‚YYYY-MM-DD
TIME	Zeit-Datentyp, ‚HH:MM:SS.S‘

Aufzählung nicht vollständig!

Constraints

- `NOT NULL` – Keine NULL-Werte

- `PRIMARY KEY` – Primärschlüssel,
  - Maximal ein Primärschlüssel pro Relation
  - Keine NULL-Werte
  - Bei einem Attribut: Deklaration direkt in der Attributliste
  - Bei mehreren Attributen: Deklaration nach den Attributen

- `UNIQUE` – Eindeutige Attributwerte
  - Es darf mehrere UNIQUE-Deklarationen geben pro Relation
  - `NULL`-Werte werden je nach DB-System akzeptiert

- Beispiel

```
CREATE TABLE Person (
  name VARCHAR(30) NOT NULL,
  geburtstag DATE NOT NULL,
  geburtsort VARCHAR(30),
  PRIMARY KEY (name, geburtstag)
)
```

Constraints

- `FOREIGN KEY` – Fremdschlüssel (referentielle Integrität)
  - kann `NULL` sein
  - Die referenzierte Attributmenge muss **`UNIQUE`** oder **`PRIMARY KEY`** sein.

- `CHECK` – Tabellen-Constraints (Wertebereich eingrenzen)

↳ (Bsp.) "Preis ist nicht negativ" CHECK (VkPreis > 0),

- Beispiel
  - ```
    CREATE TABLE Haustier (
      id INT NOT NULL PRIMARY KEY,
      name VARCHAR(30) NOT NULL,
      besitzername VARCHAR(30) NOT NULL,
      besitzergeburtstag DATE NOT NULL,
      FOREIGN KEY (besitzername, besitzergeburtstag)
            REFERENCES Person(name, geburtstag)
    )
    ```
    ```

Referentielle Integrität erzwingen

- Was soll passieren, wenn das referenzierte Tupel
  - gelöscht wird (**ON DELETE**)
  - geändert wird (**ON UPDATE**)

- Drei Varianten
  - **RESTRICT** – Verletzende Änderungen ablehnen (SQL default)
  - **CASCADE** – Kaskadierung
  - **SET NULL** – Setze Null-Werte für die Foreign Key Attribute
  - **NO ACTION** – Alias für **RESTRICT (DEFAULT)**

- Beispiel
  - ```
    CREATE TABLE Haustier (
      …,
      FOREIGN KEY(besitzername, besitzergeburtstag)
          REFERENCES Person(name, geburtstag)
              ON DELETE CASCADE)
    ```
    ```

Übung

- Erstellt folgende Tabellen:

Produkt	ProdNr	VkPreis	Bezeichnung	Abteilung
	88	200.30	Bratpfane	NullBock
	99	55.25	Schnuller	NixDa
	100	23.99	Schuhe	NixDa

Abteilung	AbtName
	NullBock
	NixDa

- Constraints:
  - „VkPreis“ darf nicht negativ werden
  - „Bezeichnung“ darf nicht NULL werden
  - Wird der „AbtName“ geändert, so soll Abteilung in Produkt auch geändert werden.
  - Eine „Abteilung“ darf nicht gelöscht werden, wenn es dazugehörige Produkte gibt

Übung

- Lösung

```
CREATE TABLE Abteilung(
    AbtName CHAR(10) PRIMARY KEY
) ENGINE=INNODB;
```

Abteilung	AbtName
	NullBock
	NixDa

Übung

- Lösung

```
CREATE TABLE Produkt(
ProdNr INT PRIMARY KEY,
VkPreis DECIMAL(6,2) NOT NULL,
Bezeichnung VARCHAR(30) NOT NULL,
Abteilung char(10) NOT NULL,
CHECK (VkPreis > 0),
FOREIGN KEY(Abteilung) REFERENCES Abteilung(AbtName) ON
UPDATE CASCADE) ENGINE=INNODB;
```

Preis ist Decim

Produkt	ProdNr	VkPreis	Bezeichnung	Abteilung
	88	200.30	Bratpfane	NullBock
	99	55.25	Schnuller	NixDa
	100	23.99	Schuhe	NixDa

Constraints nachträglich ergänzen

```
CREATE TABLE Abteilung(
AbtName CHAR(10) NOT NULL
) ENGINE=InnoDB;
```

```
CREATE TABLE Produkt(
ProdNr INT PRIMARY KEY,
VkPreis DECIMAL(6,2) NOT NULL,
Bezeichnung VARCHAR(30) NOT NULL,
Abteilung char(10) NOT NULL) ENGINE=InnoDB;
```

Primärschlüssel für „AbtName“

```
ALTER TABLE Abteilung
ADD CONSTRAINT pk_abteilung PRIMARY KEY(AbtName);
```

Fremdschlüssel für „Abteilung“ der Tabelle Produkt

```
ALTER TABLE Produkt
ADD CONSTRAINT fk_abteilung FOREIGN KEY(Abteilung)
REFERENCES Abteilung(AbtName) ON UPDATE CASCADE;
```

CHECK-Constraint

```
ALTER TABLE Produkt
ADD CONSTRAINT ck_produkt_vkPreis CHECK (VkPreis > 0);
```

INSERT – Datensätze einfügen

```
INSERT INTO Tabellenname
VALUES (value1, value2, value3,...)

INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

Beispiel:

```
INSERT INTO Abteilung VALUES ('NullBock'), ('NixDa');
```

Abteilung	AbtName
	NullBock
	NixDa

UPDATE, DELETE – Datensätze ändern/löschen

- UPDATE (Datensätze, die die WHERE-Klausel erfüllen, ändern)

```
UPDATE Tabellenname
SET Attribut = neuerWert
WHERE Attribut = alterWert;
```

Beispiel:

```
UPDATE Abteilung
SET AbtName = 'Sowieso'
WHERE name = 'NullBock';
```

- DELETE (Datensätze löschen)

```
DELETE FROM Tabellenname
WHERE Spaltenname = 'Wert'
```

Beispiel:

```
DELETE FROM Person
WHERE AbtName = 'NixDa';
```

Weitere DDL-Anweisungen

- Neue Spalte „EinkaufsPreis“ der Tabelle Produkt hinzufügen

```
ALTER TABLE Produkt ADD EinkaufsPreis DECIMAL(6,2);
```

- Umbenennen eines Attributes

```
ALTER TABLE Produkt CHANGE EinkaufsPreis EinkPreis
DECIMAL(6,2);
```

- Typ eines Attributs ändern

```
ALTER TABLE Produkt MODIFY Bezeichnung VARCHAR(99);
```

- Attribut löschen

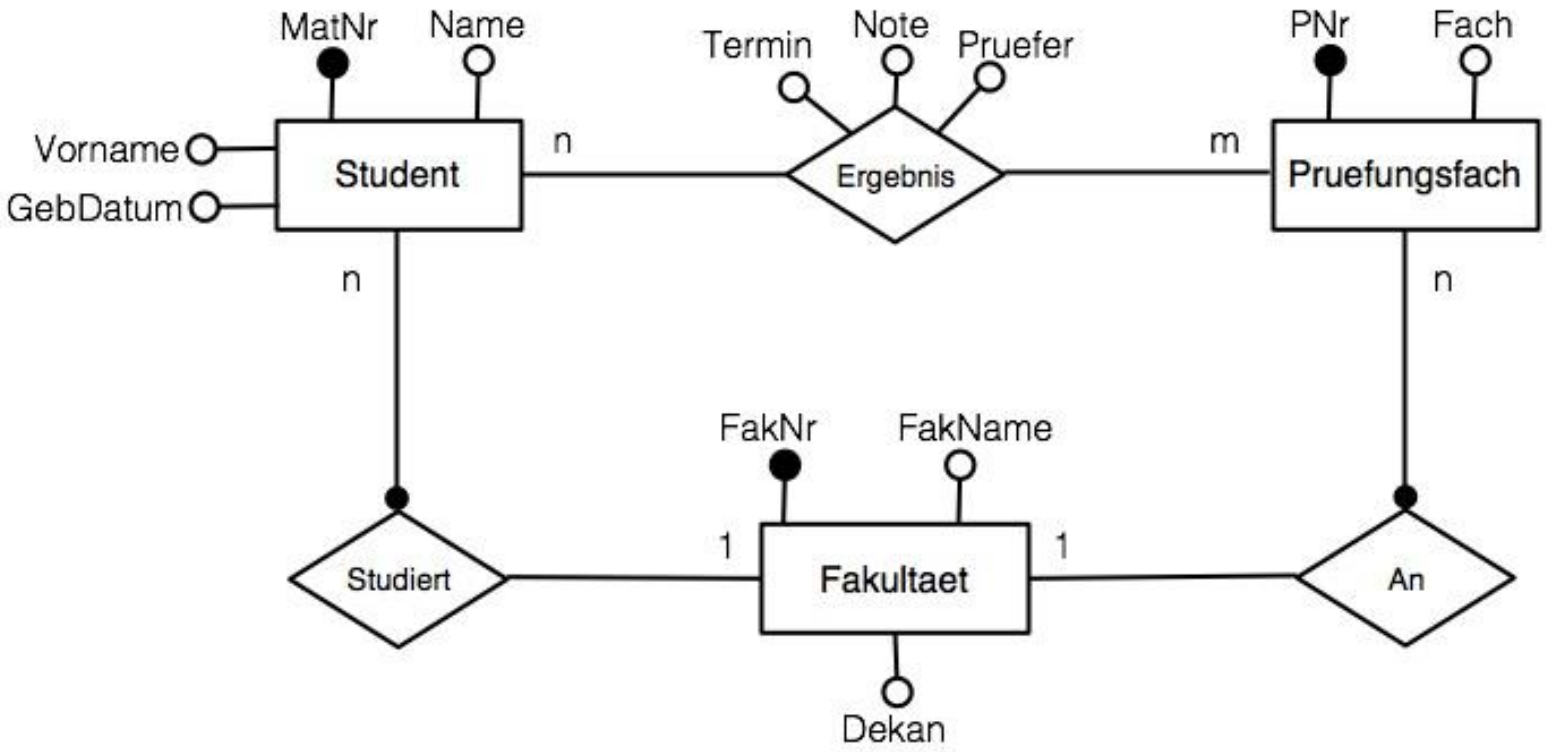
```
ALTER TABLE Produkt DROP EinkPreis;
```

- Tabelle löschen

```
DROP TABLE Produkt;
```

Aufgabe 1

- Erstellen Sie das Relationenschema sowie dann den SQL-DDL code zum Anlegen der Tabellen nach dem vorliegenden E/R-Typ-Diagramm (Entwurfsentscheidungen für das Schema!).



MySQL Referenzhandbuch
http://dev.mysql.com/doc/refman/5.1/de/

Aufgabe 1 – Lösung

- Erstellen Sie das Relationenschema sowie dann den SQL-DDL code zum Anlegen der Tabellen nach dem vorliegenden E/R-Typ-Diagramm (Entwurfsentscheidungen für das Schema!).

- Relationen Schema:
  - Student (MatNr, Name, Vorname, GebDatum, *FakNr → Fakultaet*)
  - Pruefungsfach (PNr, Fach, *FakNr → Fakultaet*)
  - Ergebnis (*MatNr → Student*, *PNr → Pruefungsfach*, Note, Pruefer, Termin)
  - Fakultaet (FakNr, FakName, Dekan)

Aufgabe 1 – Lösung

- Tabellen anlegen

```
CREATE TABLE Fakultaet (
FakNr INT PRIMARY KEY,
FakName VARCHAR(10) NOT NULL,
Dekan VARCHAR(30)
) ENGINE=INNODB;


CREATE TABLE Pruefungsfach (
PNr INT PRIMARY KEY,
Fach VARCHAR(10) NOT NULL,
FakNr INT NOT NULL,
FOREIGN KEY(FakNr) REFERENCES Fakultaet(FakNr) ON DELETE
CASCADE) ENGINE=INNODB;
```

Aufgabe 1 – Lösung

- Tabellen anlegen:

```
CREATE TABLE Student(
MatNr INT PRIMARY KEY,
Name VARCHAR(30) NOT NULL,
Vorname VARCHAR(20) NOT NULL,
GebDatum DATE NOT NULL,
FakNr INT NOT NULL,
FOREIGN KEY (FakNr) REFERENCES Fakultaet(FakNr) ON DELETE
   CASCADE) ENGINE=INNODB;
```

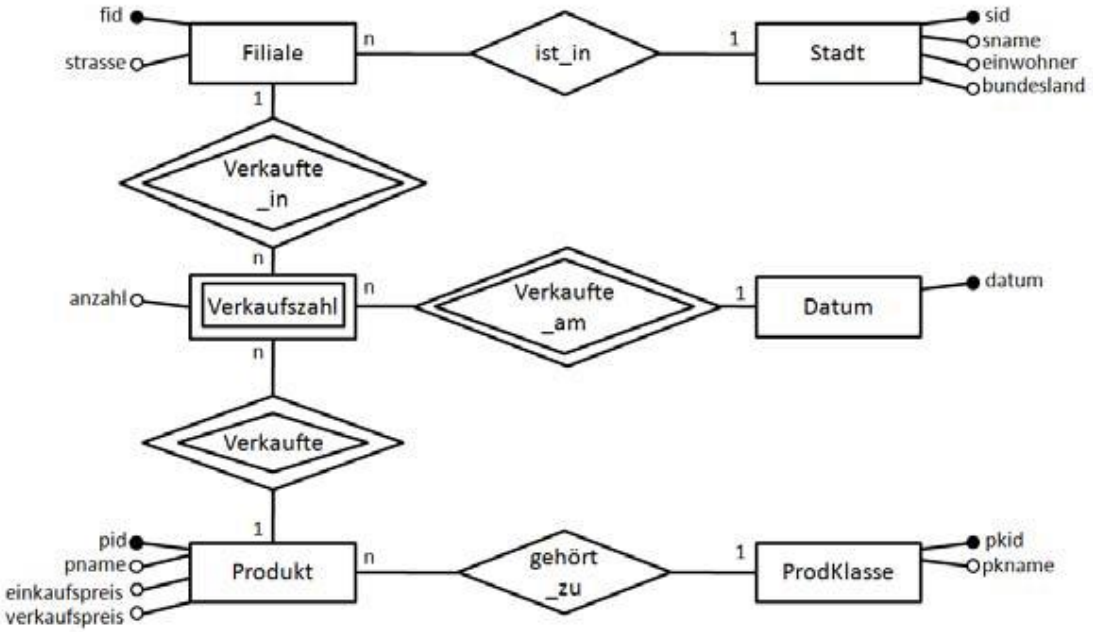
Aufgabe 1 – Lösung

- Tabellen anlegen:

```
CREATE TABLE Ergebnis(
MatNr INT NOT NULL,
PNr INT NOT NULL,
Pruefer VARCHAR(30),
Termin DATE,
Note DECIMAL(3,1) CHECK (NOTE IN(1.0, 1.3, 1.7, 5.0)),
PRIMARY KEY (MatNr, PNr),
FOREIGN KEY (MatNr) REFERENCES Student(MatNr) ON DELETE CASCADE,
FOREIGN KEY (PNr) REFERENCES Pruefungsfach(PNr) ON DELETE CASCADE
   ) ENGINE = INNODB;
```

ACHTUNG: „... CHECK(NOTE IN(... „Sollte um die weitere Noten ergänzt werden

Aufgabe 2



- Durchschnittliche Einnahmen (ohne zeitliche Einschränkung) der Filialen pro Bundesland, absteigend sortiert nach Einnahmen: Die Einnahmen berechnen sich aus den Verkaufszahlen der Produkte (Anzahl * Verkaufspreis). Ausgegeben werden das Bundesland und die durchschnittlichen Einnahmen, absteigend nach durchschnittlichen Einnahmen sortiert.

Aufgabe 2 - Lösung

```
SELECT s.bundesland, AVG(x.summe) AS avgsumme
FROM Filiale f, Stadt s,
   (
   -- Summierten Einnahmen pro Filiale
   SELECT v.filiale, sum(p.verkaufspreis * v.anzahl) as summe
   FROM  produkt p, verkaufszahl v
   WHERE v.produkt = p.pid
   GROUP BY v.filiale
) x
WHERE f.fid = x.filiale AND f.stadt = s.sid
GROUP BY s.bundesland
ORDER BY avgsumme DESC;
```

Hilfreiche Links

- MySQL 5.1 Referenzhandbuch(Deutsch)
  http://dev.mysql.com/doc/refman/5.1/de/
- w3schools
  http://www.w3schools.com/sql/