

# LE 8 – IBIS – Datenbanken

## INSERT, UPDATE, DELETE und Data Definition Language

Angelehnt an Skript „Datenmodellierung und Datenmanagement“, THM  
Mitwirkende Autoren: Prof. Dr. Guckert; Timo Péus, Dr. Thomas Farrenkopf,  
Melanie Vanderpuye, Prof. Dr. Grüne (2017)

Prof. Dr. Markus Grüne, FB03, Wirtschaftsinformatik

# Inhalt

- Insert
- Update
- Delete

## Lernziel / Fragen

SQL-Abfragen, die Änderungen am Datenbankzustand bewirken.

Wie füge ich neue Tupel in eine Tabelle ein?  
Wie kann ich Tupel ändern? Wie löschen?

# INSERT

Im Folgenden werden drei Varianten für das INSERT-Statement dargestellt.

Einige MySQL-Spezifika werden wir hier ignorieren.

Die erste Form ist die allgemeinste, die zweite erlaubt es mehrere Zeilen mit einem Insert einzufügen, die dritte kann die Zeilen aus einer anderen Tabelle lesen.

Wird für eine Spalte kein Wert angegeben, so wird sie wenn möglich auf Null oder auf einen Default-Wert gesetzt (wenn die Spalte so definiert wurde. Andernfalls gibt es einen Fehler!

# INSERT I (MySQL)

INSERT INTO *tabelle*

[(*col\_name*,...)]

VALUES ({*expr* | DEFAULT},...);

```
INSERT INTO lehrer
(Nachname, Vorname, gebdatum,
schuleintritt, stufe)
VALUES
('Herberger', 'Sepp', '1958-04-
01', '1983-08-01',
'Oberstudienrat')
```

AUTOINC  
NULLABLE

Column Name	Datatype	NOT NULL	AUTO INC
PNr	INTEGER	✓	✓
Titel	CHAR(20)		
Nachname	CHAR(20)	✓	
Vorname	CHAR(20)	✓	
Gebdatum	DATE	✓	
SchulEintritt	DATE	✓	
Stufe	CHAR(20)	✓	

Reihenfolge der Spalten spielt keine Rolle – sie muss nur zu den Werten passen!

## INSERT II (MySQL)

INSERT INTO *tbl\_name*

[(*col\_name*,...)]

VALUES ({*expr* | DEFAULT},...),

(...),... ;

```
INSERT INTO lehrer
(Nachname,Vorname,gebdatum,schuleintritt,stufe)
VALUES
('Herberger','Sepp','1958-04-01','1983-08-01',
'Oberstudienrat'),
('Hörbiger','Christine','1959-05-01','1983-08-01',
'Oberstudienrätin');
```

## INSERT III

```
INSERT INTO tbl_name  
  [(col_name,...)]  
SELECT ...
```



Zugriff auf eine andere Tabelle!

# Update

Aktualisierungen vorhandener Einträge erfolgen mit dem Update-Befehl.

**Vorsicht bei der Where-Klausel: Die komplette Selektion wird aktualisiert!**

```
UPDATE tbl_name  
  SET col_name1=expr1 [, col_name2=expr2 ...]  
[WHERE where_definition]
```



## Beförderung

```
UPDATE lehrer  
SET stufe='Oberstudienrat'  
WHERE stufe='Studienrat';  
UPDATE lehrer  
SET stufe='Oberstudienrätin'  
WHERE stufe='Studienrätin';
```

## Damit gehen beide auf einmal

```
UPDATE lehrer  
SET stufe=concat('Ober',LCASE(stufe))  
WHERE stufe IN ('Studienrat','Studienrätin')
```

# DELETE

Gelöscht werden Sätze mit dem Delete-Befehl.

**Vorsicht:**

**Auch Delete arbeitet mengenorientiert über die Where-Klausel.**

```
DELETE  
FROM tbl_name  
[WHERE where_definition]
```

```
DELETE FROM lehrer;
```

Löscht alle Lehrer

# Übung

Frau Ottilie Nerz macht Fortbildungen und kann daher das Amt der Klassenlehrerin nicht mehr ausüben. Kollegin Laura Lukas übernimmt ihre Klassen.

Außerdem fängt am 1.8.2005 eine neue Kollegin an. Sie heißt Angelika Ferkel. Ihr Geburtsdatum hat sie leider noch nicht gemeldet. Arbeiten Sie in einer Transaktion und „commiten“ Sie die Änderungen!

Löschen Sie alle Schüler der Klasse mit KNr 8

Machen Sie diese Änderungen rückgängig.

## Lernziel / Fragen

SQL-Abfragen, die Änderungen am Datenbankzustand bewirken.

Wie füge ich neue Tupel in eine Tabelle ein?

Wie kann ich Tupel ändern? Wie löschen?

# Inhalt – DDL Data Definition Language

- Sprachkonstrukte für die Definition von Datenbankobjekten

## Anlegen von Datenbankobjekten mit DDL

Wie lege ich Tabellen an?

Wie erzeuge ich eine View?

Wie erzeuge ich einen Index?

Wie definiere ich die Regeln für die referentielle Integrität?

# Datenbank

MySQL ist eine Datenbankmanagementsystem, das viele so genannten Datenbanken (databases) verwalten kann.

Dies sind logische Zusammenfassungen von zusammengehörigen Daten.

Mit `create database <dbname>` erzeugt man eine solche (wenn man darf). Die Datenbank wird auch als Schema angesprochen.

Mit `use <dbname>` kann man eine vorhandene Datenbank nutzen (wenn man darf).

Andere Datenbanken verwenden den Begriff Schema für die Tabellen eines Benutzers (Owners).

# Create und Alter

Tabellen und weitere Datenbankobjekte werden mit entsprechenden **Create-Befehlen** erstellt. Beim Create Table werden alle Attribute mit Wertebereichen und evtl. weiteren Informationen angegeben. Außerdem werden hier weitere Details für die physische Umsetzung der Tabelle angegeben!

Mit dem Befehl **Alter Table** können Tabellen nachträglich geändert werden (neue Spalten,...). Hierbei sind bestimmte Regeln zu beachten.

Eine Tabelle ist bereits gefüllt. Sie brauchen eine neue Spalte, die nicht *nullable* ist. Welche Schritte sind wohl notwendig?



# Create Table I

Name der Datenbank/des Schemas      Name der Tabelle

```
CREATE TABLE `schule`.`ABC` (  
    `col1` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT  
        COMMENT 'Spalte 1',  
    `col2` VARCHAR(45) NOT NULL  
        COMMENT 'Spalte 2',  
    PRIMARY KEY(`col1`)  
)  
ENGINE = InnoDB  
COMMENT = 'Ein Kommentar';
```

Spaltendetails

Kommentar zur Spalte

Definition des PK

Kommentar zur Tabelle

Angabe von Default-Werten ist möglich!

# Referenzielle Integrität

Wir haben die Notwendigkeit für die Normalisierung von Datenmodellen aufgrund der verschiedenen Anomalien erkannt.

Zur Lösung werden Informationen auf mehrere Relationen/Tabellen verteilt.

Verweise zwischen den Tabellen sorgen für konsistente Datenmodelle (Sätze, die referenziert werden, müssen in der „fremden“ Tabelle vorhanden sein).

Dies erfolgt über so genannte **Fremdschlüssel** (Foreign Keys).

Diese **referentiellen Constraints** stellen die referentielle Integrität der Datenbank sicher (kein Verweis zeigt ins Leere).

Spalten, die auf andere Tabellen verweisen dürfen Null-Werte enthalten, wenn die Attributdefinition dies zulässt.

# Create Table II

```
CREATE TABLE `unterricht` (  
  `KNr` int(11) NOT NULL,  
  `FNr` int(11) NOT NULL,  
  `PNr` int(11) NOT NULL,  
  PRIMARY KEY (`KNr`, `FNr`),  
  KEY `IX_Lehrer_gibt_Fach` (`PNr`),  
  KEY `IX_Fach_in_Klasse` (`KNr`),  
  KEY `IX_wird_unterrichtet` (`FNr`),  
  CONSTRAINT `unterricht_ibfk_1` FOREIGN KEY (`PNr`)  
    REFERENCES `lehrer` (`PNr`),  
  CONSTRAINT `unterricht_ibfk_2` FOREIGN KEY (`KNr`)  
    REFERENCES `klasse` (`KNr`),  
  CONSTRAINT `unterricht_ibfk_3` FOREIGN KEY (`FNr`)  
    REFERENCES `fach` (`FNr`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Index für Foreign Key

Default=Restrict

# Wie werde ich eine Tabelle wieder los?

Mit dem Drop-Befehl kann eine Tabelle wieder gelöscht werden. Dies geht nur, wenn die Tabelle nicht Ziel von Foreign Key Verweisen ist.

```
DROP TABLE <Tabellenname>;
```

## Änderung am Tabellenschema

Das ALTER-Statement lässt Änderungen an bestehenden Tabellen zu. Dabei sind nur Änderungen erlaubt, die zu keinem Informationsverlust führen.

Bestimmte Voraussetzungen müssen fallweise erfüllt sein (z.B. neue Spalte an Tabelle, die bereits Einträge enthält muss NULLABLE sein).

Fügt eine neue Spalte an die Tabelle *schueler* an.

```
alter table schueler add groesse decimal(8,3);
```

## Alter Table

```
ALTER TABLE `schule`.`fach`  
  ADD COLUMN `fachleiter` INTEGER UNSIGNED NOT NULL  
    AFTER `Bezeichnung`,  
  ADD INDEX `IX_Fachleiter` (`fachleiter`),  
  ADD CONSTRAINT `FK_fachleiter` FOREIGN KEY  
    `FK_fachleiter` (`fachleiter`)  
      REFERENCES `lehrer` (`PNr`)  
    ON DELETE RESTRICT  
    ON UPDATE RESTRICT;
```

# DDL

Für die verschiedenen Datenbankobjekte gibt es Create-Statements mit aufwändiger Syntax, die es jeweils erlaubt alle Details der Objekte zu spezifizieren (-> Handbuch bzw. Literatur).

Einfache Kopien von Tabellen können mit Hilfe eines Select-Statements erstellt werden:

```
CREATE TABLE NEU AS SELECT * FROM ALT;
```

Dabei werden die Daten gleich mit kopiert.

Wie erzeugt man eine leere Kopie?

# Indices

Indexe sollen primär einen effizienten Zugriff auf die Daten ermöglichen.

Durch einen eindeutigen Schlüssel (UNIQUE INDEX) kann implizit ein Constraint für die Tabelle definiert: Das Attribut bzw. die Kombination der Attribute im Schlüssel muss eindeutig sein.

Indices werden häufig für Spalten

- die zum Primärschlüssel
  - zu einem Fremdschlüssel
- gehören automatisch erzeugt.



## Create Index

```
CREATE [UNIQUE|...] INDEX index_name ON tbl_name  
(index_col_name,...)
```

```
DROP INDEX index_name;
```

# Übung

Fügen Sie der Tabelle *Fach* ein Feld *Fachleiter* vom Typ Integer hinzu!

Erzeugen Sie eine Tabelle *LehrerCopy*, die alle Datensätze aus der Tabelle *Lehrer* enthält!

Erzeugen Sie eine View *LehrerView*, die alle Datensätze aus der Tabelle *Lehrer* darstellt!

Wo ist der Unterschied?

Erstellen Sie eine View, die die Informationen der Tabelle Schüler anzeigt. Anstelle der *Knr* soll dort aber die Bezeichnung der Klasse angezeigt werden.

# Übung

Fügen Sie der Tabelle *Fach* ein Feld *Fachleiter* vom Typ Integer hinzu!

```
ALTER TABLE fach ADD fachleiter INTEGER;
```

Erzeugen Sie eine Tabelle *LehrerCopy*, die alle Datensätze aus der Tabelle *Lehrer* enthält!

```
CREATE TABLE lehrercopy AS SELECT * FROM lehrer;
```

Erzeugen Sie eine View *LehrerView*, die alle Datensätze aus der Tabelle *Lehrer* darstellt!  
Wo ist der Unterschied?

```
CREATE view lehrerview AS SELECT * FROM lehrer;
```

# Übung

Erstellen Sie eine View, die die Informationen der Tabelle Schüler anzeigt. Anstelle der *Knr* soll dort aber die Bezeichnung der Klasse angezeigt werden.

```
CREATE OR REPLACE VIEW schuelerinfo AS
  SELECT s.snr, s.nachname, s.vorname,
         s.gebdatum, s.schuleintritt, k.bezeichnung
  FROM   schueler s JOIN klasse k
  ON     (s.knr=k.knr);
```

# Lernziel / Fragen

Wie lege ich Tabellen an?

Wie erzeuge ich eine View?

Wie erzeuge ich einen Index?

Wie definiere ich die Regeln für die referentielle Integrität?