

# SQL

## SQL: Structured Query Language

Früherer Name: SEQUEL

Standardisierte Anfragesprache für relationale DBMS:  
SQL-89, SQL-92, SQL-99

SQL ist eine **deklarative** Anfragesprache

# Teile von SQL

---

Vier große Teile:

- DRL: Data Retrieval Language
- DML: Data Manipulation Language
- DDL: Data Definition Language
- DCL: Data Control Language

# DRL

DRL enthält Kommandos für Anfragen

Einfache Anfrage besteht aus den drei Klauseln

**select, from** und **where**

**select** *Liste von Attributen*  
**from** *Liste von Relationen*  
**where** *Prädikat,*

# DML

DML enthält Befehle um

- Daten einzufügen
- Daten zu löschen
- Daten zu ändern

**insert, update, delete**

# DDL

- Mit Hilfe der DDL kann das Schema einer Datenbank definiert werden
- Enthält auch Befehle, um den Zugriff auf Daten zu kontrollieren

**create table, alter table, create view, create index**

- entsprechend auch Löschoperationen **drop ..**

# DCL

- Enthält Befehle um den Fluss von Transaktionen zu steuern
- Eine Transaktion ist eine Menge von Interaktionen zwischen Anwendung/Benutzer und dem DBMS
- Wird später im Rahmen von Transaktionsverwaltung behandelt

# Varianten von SQL

---

- Eine Datenbank kann nicht nur interaktiv benutzt werden
- SQL kann in andere Programmiersprachen eingebettet werden
- Problem: SQL ist mengenorientiert, die meisten Programmiersprachen nicht

# Embedded SQL

- SQL-Befehle werden direkt in die jeweilige Hostsprache eingebettet (z.B. C, C++, Java, etc.)
- SQL-Befehle werden durch ein vorangestelltes **EXEC SQL** markiert
- Sie werden vom Präprozessor durch Konstrukte der jeweiligen Sprache ersetzt

# Dynamic SQL

- Wird eingesetzt, wenn die Anfragen zur Übersetzungszeit des Programms noch nicht bekannt sind
- Standardisierte Schnittstellen
  - ODBC (Open Database Connectivity)
  - JDBC (für Java)
- Flexibler, aber üblicherweise etwas langsamer als Embedded SQL

# DDL: Create Table Statement

Syntaxdiagramm umspannt viele Seiten!!

Einfache Form:

```
CREATE TABLE Tabellename (  
    Attribut_1 Datentyp_1 [NOT NULL],  
    ...  
    Attribut_n Datentyp_n [NOT NULL]);
```

# DDL: Beispiel Create Table Statement

```
CREATE TABLE Professoren  
  (PersNr INTEGER NOT NULL,  
   Name VARCHAR(30) NOT NULL,  
   Rang CHAR(2),  
   Raum INTEGER);
```

```
CREATE TABLE Vorlesungen  
  (VorlNr INTEGER NOT NULL,  
   Titel VARCHAR(30),  
   SWS INTEGER,  
   gelesenVon INTEGER);
```

# DDL: Datentypen in SQL - Zahlen

- **VARCHAR** (n) variable Zeichenkette der maximalen Länge n
- **CHAR[ACTER]** (n) feste Zeichenkette von n Byte
- **NUMERIC** [(p[, s])] vorzeichenbehaftete Zahl mit insgesamt p Stellen, davon s hinter dem Komma  
auch **DEC[IMAL]** [(p,s)] (erlaubt Abspeicherung größerer Werte)
- **INT[EGGER]** vorzeichenbehaftete ganze Zahl
- **SMALLINT** wie INTEGER, kleinerer Wertebereich
- **FLOAT** [(p)] (gerundete) Gleitkommazahl (mind. p Bits Präzision)  
**REAL, DOUBLE PRECISION** Abkürzungen für FLOAT(p) mit implementierungsabhängigen Werten für p

# DDL: Datentypen in SQL – Datum und Zeit

- **DATE** Gültiges Datum
- **TIME** Zeit (von 00:00:00 bis 23:59:59)
- **TIMESTAMP** Zeitstempel (Kombination Datum und Zeit)

(ORACLE hat nur DATE und nutzt diesen als Zeitstempel)

# DDL: Datentypen in SQL – Zeichenketten, Binärdaten

- **LONG** variable Zeichenkette bis zu 2 GB (**TEXT** SQL Server)
- **CLOB** Zeichenketten bis 4 GB
  
- **RAW** (n) Binärdaten der Länge n, n zwischen 1 und 2000 Bytes
- **LONG RAW** Binärdaten bis zu 2 GB
- **BLOB** Binärdaten bis 4 GB
  
- **CFILE, BFILE** Zeiger auf Dateien (Text, Binär) (Oracle)
- **DATALINK** Zeiger auf Datei (DB2)
- **MONEY / SMALLMONEY** (SQL Server)
- ...

eingeschränkte Operationen!

# DDL: Integritätsbedingungen

---

- Zu den Aufgaben eines DBMS gehört es auch, die Konsistenz der Daten zu sichern
- Semantische Integritätsbedingungen beschreiben Eigenschaften der modellierten Miniwelt
- DBMS kann mit Hilfe von Constraints automatisch diese Bedingungen überprüfen

# Primärschlüsselbedingung

---

Attribut oder Attributkombination hat in jeder Ausprägung der Relation keinen Wert, der mehr als einmal vorkommt

# DDL: Primärschlüsselbedingung

```
CREATE TABLE Tabellename (  
    Attribut_1 Datentyp_1 [NOT NULL],  
    ...  
    Attribut_n Datentyp_n [NOT NULL]),  
[CONSTRAINT constraintname_pk] PRIMARY KEY  
    (Attribut_i, ...,Attribut_p));
```

# DDL: Beispiel Primärschlüssel

```
CREATE TABLE Professoren
  (PersNr INTEGER NOT NULL,
   Name VARCHAR(30) NOT NULL,
   Rang CHAR(2),
   Raum INTEGER,
   PRIMARY KEY (PersNr) );
```

```
CREATE TABLE Vorlesungen
  (VorlNr INTEGER NOT NULL,
   Titel VARCHAR(30),
   SWS INTEGER,
   gelesenVon INTEGER,
   PRIMARY KEY (VorlNr));
```

# DDL: Primärschlüsselbedingung (Kurzform)

```
CREATE TABLE Professoren  
  (PersNr INTEGER NOT NULL PRIMARY KEY,  
   Name VARCHAR(30) NOT NULL,  
   Rang CHAR(2),  
   Raum INTEGER);
```

```
CREATE TABLE Vorlesungen  
  (VorlNr INTEGER NOT NULL PRIMARY KEY,  
   Titel VARCHAR(30),  
   SWS INTEGER);
```

# DDL: Weitere Integritätsbedingungen

Neben Primärschlüsseln gibt es eine ganze Reihe weiterer Integritätsbedingungen:

- NOT NULL
- Unique
- check-Klauseln

# DDL: NOT NULL Bedingung

- Erzwingt definierte Attributwerte beim Einfügen von Tupeln
- Zwingend für Schlüssel
- Default-Angabe möglich

```
CREATE TABLE defaults (  
  Id INTEGER NOT NULL PRIMARY KEY,  
  Ort VARCHAR(80) DEFAULT „GARCHING“,  
  Alter SMALLINT DEFAULT 20,  
  Groesse SMALLINT NOT NULL);
```

# DDL: UNIQUE Bedingung

Stellt für Attribut Schlüsseleigenschaft sicher

```
CREATE TABLE Professoren  
(PersNr INTEGER PRIMARY KEY,  
Name VARCHAR(30) NOT NULL,  
Rang CHAR(2) CHECK (Rang in ('C2', 'C3', 'C4')),  
Raum INTEGER NOT NULL UNIQUE);
```

# DDL: Check Klauseln

- Durch **check**-Klauseln kann der Wertebereich für Attribute eingeschränkt werden
- Beispiel:

```
CREATE TABLE Professor (  
  PersNr INTEGER NOT NULL PRIMARY KEY,  
  Name VARCHAR(80) NOT NULL,  
  Raum INTEGER  
  CHECK (Raum > 0 AND Raum < 99999));
```

# Referentielle Integrität

- $R$  und  $S$  sind zwei Relationen mit den Schemata  $R$  bzw.  $S$
- $k$  ist Primärschlüssel von  $R$
- Dann ist  $f \in S$  ein Fremdschlüssel, wenn für alle Tupel  $s \in S$  gilt:
  - $s.f$  enthält entweder nur Nullwerte oder nur Werte ungleich Null
  - Enthält  $s.f$  keine Nullwerte, so existiert ein Tupel  $r \in R$  mit  $s.f = r.k$
- Die Einhaltung dieser Eigenschaften wird *referentielle Integrität* genannt

# DDL: Fremdschlüsselbedingung

```
CREATE TABLE Tabellename (  
    Attribut_1 Datentyp_1 [NOT NULL],  
    ...  
    Attribut_n Datentyp_n [NOT NULL]),  
  
[CONSTRAINT constraintname_pk] PRIMARY KEY  
    (Attribut_i, ..., Attribut_p),  
  
CONSTRAINT constraintname_fk FOREIGN KEY  
(Attribut_j, ..., Attribut_l) REFERENCES  
Elterntabellename (Attribut_t, ..., Attribut_v));
```

# DDL: Beispiel Fremdschlüssel

```
CREATE TABLE Professoren
  (PersNr INTEGER NOT NULL,
   Name VARCHAR(30) NOT NULL,
   Rang CHAR(2),
   Raum INTEGER,
   PRIMARY KEY (PersNr) );

CREATE TABLE Vorlesungen
  (VorlNr INTEGER NOT NULL,
   Titel VARCHAR(30),
   SWS INTEGER,
   gelesenVon INTEGER,
   PRIMARY KEY (VorlNr),
   CONSTRAINT gelesen_fk FOREIGN KEY
   (gelesen_von) REFERENCES Professoren (PersNr) );
```

# DDL: Beispiel Fremdschlüssel (Kurzform)

```
CREATE TABLE Professoren  
  (PersNr INTEGER NOT NULL PRIMARY KEY,  
   Name VARCHAR(30) NOT NULL,  
   Rang CHAR(2),  
   Raum INTEGER);
```

```
CREATE TABLE Vorlesungen  
  (VorINr INTEGER NOT NULL PRIMARY KEY,  
   Titel VARCHAR(30),  
   SWS INTEGER,  
   gelesenVon INTEGER REFERENCES Professoren);
```

# DDL: Fremdschlüssel Varianten

- Änderungen an Schlüsselattributen können automatisch propagiert werden
- set null: alle Fremdschlüsselwerte, die auf einen Schlüssel zeigen, der geändert oder gelöscht wird, werden auf NULL gesetzt
- cascade: alle Fremdschlüsselwerte, die auf einen Schlüssel zeigen, der geändert oder gelöscht wird, werden ebenfalls auf den neuen Wert geändert bzw gelöscht

# DDL: Beispiel Fremdschlüssel Varianten

```
CREATE TABLE Vorlesungen
  (VorINr INTEGER NOT NULL PRIMARY KEY,
  Titel VARCHAR(30),
  SWS INTEGER,
  gelesenVon INTEGER REFERENCES Professoren
  ON DELETE SET NULL);
```

```
CREATE TABLE hoeren
  (MatrNr INTEGER REFERENCES Studenten
  ON DELETE CASCADE,
  VorINr INTEGER REFERENCES Vorlesungen
  ON DELETE CASCADE,
  PRIMARY KEY (MatrNr, VorINr));
```