

Relationales Modell: SQL-DDL

SQL-DDL umfasst alle Klauseln von SQL, die mit Definition von

- Typen
- Wertebereichen
- Relationenschemata
- Integritätsbedingungen

zu tun haben

SQL als Definitionssprache

- *Externe Ebene*
 - ◆ **create view**
 - ◆ **drop view**
- *Konzeptuelle Ebene*
 - ◆ **create table**
 - ◆ **alter table**
 - ◆ **drop table**

7. Datenbankdefinitionssprachen

- ➡ **SQL-DDL** Teil der Standardsprache für relationale Datenbanksysteme: SQL
- ➡ **CODASYL-DDL**: Netzwerkmodell
- ➡ **IMS-DDL**: hierarchisches Modell
- ➡ **ODL** (Object Definition Language) für objektorientierte Datenbanksysteme nach dem ODMG-Standard

Anforderungen an eine relationale DDL

- nach Codd 1982 Sprachmittel zur Definition von
 1. Attributen
 2. Wertebereichen
 3. Relationenschemata
 4. Primärschlüsseln
 5. Fremdschlüsseln
- Praxis SQL-89: Relationenschemata mit
 - ◆ Attributen und
 - ◆ Wertebereichen

Die Anweisung create table

```
create table basisrelationenname  
  (spaltenname_1 wertebereich_1 [not null],  
   ...  
  spaltenname_k wertebereich_k [not null])
```

Die Anweisung create table II

Mit **not null** können in bestimmten Spalten *Nullwerte* als Attributwerte ausgeschlossen werden:

```
create table Bücher  
  ( ISBN char(10) not null,  
    Titel varchar(200),  
    Verlagsname varchar(30) )
```

SQL als Definitionssprache II

■ Konzeptuelle Ebene (SQL-92)

- ◆ **create domain**
- ◆ **alter domain**
- ◆ **drop domain**

■ Interne Ebene

- ◆ **create index**
- ◆ **alter index**
- ◆ **drop index**

Erlaubte Wertebereiche in create table

- *integer* (oder auch *integer4*, *int*), *smallint* (oder auch *integer2*)
- *float(p)* (oder auch kurz *float*)
- *decimal(p,q)* und *numeric(p,q)* mit jeweils *q* Nachkommastellen
- *character(n)* (oder kurz *char(n)*, bei $n = 1$ auch *char*) für Strings fester Länge *n*
- *character varying(n)* (oder kurz *varchar(n)*) für Strings variabler Länge bis zur Maximallänge *n*
- *bit(n)* oder *bit varying(n)* analog für Bitfolgen
- *date*, *time* bzw. *timestamp* für Datums-, Zeit- und kombinierte Datums-Zeit-Angaben

Beispiel Tabellendefinition mit IEF

```
create table Bücher
( ISBN char(10) not null,
  Titel varchar(200),
  Verlagsname varchar(30),
  primary key (ISBN),
  foreign key (Verlagsname)
    references Verlage (Verlagsname) )
```

Erweiterungen in SQL-92

Neben Primär- und Fremdschlüssel in SQL-92:

- **default**-Klausel: Defaultwerte für Attribute
- **create domain**-Anweisung benutzerdefinierte Wertebereiche
- **check**-Klausel weitere lokale Integritätsbedingungen innerhalb der zu definierenden Wertebereiche, Attribute und Relationenschemata

SQL-89 Level 2 mit IEF

- zweite Stufe der SQL-89-Norm sieht Zusatz IEF (Integrity Enhancement Feature) vor
- Definition von Schlüsseln und Fremdschlüsseln

create table in SQL-92

```
create table Bücher
( ISBN char(10),
  Titel varchar(200),
  Verlagsname varchar(30),
  primary key (ISBN),
  foreign key (Verlagsname)
    references Verlage (Verlagsname) )
```

not null implizit durch die **primary key**-Klausel

Integritätsbedingungen mit check-Klausel I

```
create domain Gebiete varchar(20) default 'Informatik'  
check ( value in ( 'Informatik', 'Mathematik',  
                  'Elektrotechnik', 'Linguistik' ) )
```

Integritätsbedingungen mit check-Klausel III

```
create table Buch_Versionen (  
  ISBN char(10),  
  Auflage smallint check(Auflage > 0),  
  Jahr integer check(Jahr between 1800 and 2020),  
  Seiten integer check(Seiten > 0),  
  Preis decimal(8,2) check(Preis ≤ 250),  
  primary key (ISBN, Auflage),  
  foreign key (ISBN) references Bücher (ISBN),  
  check((select sum(Preis) from Buch_Versionen) <  
        (select sum(Budget) from Lehrstühle))
```

Definition eines Wertebereichs

```
create domain Gebiete varchar(20) default 'Informatik'
```

```
create table Vorlesungen (  
  V_Bezeichnung varchar(80) not null,  
  SWS smallint,  
  Semester smallint,  
  Studiengang Gebiete )
```

```
create table Mitarbeiter (  
  PANr integer not null,  
  AngNr char(10) not null,  
  Fachbereich Gebiete,  
  Gehalt decimal(10,2),  
  Raum integer,  
  Einstellung date )
```

Integritätsbedingungen mit check-Klausel II

```
create table Vorlesungen (  
  V_Bezeichnung varchar(80) not null, primary key,  
  SWS smallint check(SWS ≥ 0),  
  Semester smallint check(Semester between 1 and 9),  
  Studiengang Gebiete )
```

alter table-Kommando in SQL-2

Statt

```
add spaltenname wertebereich
```

auch Angabe von Default-Werten und **check**-Klauseln erlaubt:

```
add Budget decimal(8,2) default 10000
  check (Budget > Anzahl_Planstellen * 1000)
```

Die Anweisung drop table

```
drop table basisrelationenname { restrict | cascade }
```

restrict und **cascade** analog zum **drop** bei Attributen

Die Anweisungen alter table und drop table

- Syntax des **alter table**-Kommandos in SQL-89:

```
alter table basisrelationenname
  add spaltenname wertebereich
```

```
alter table Lehrstühle
  add Budget decimal(8,2)
```

- Wirkung ist:

- ◆ Änderung des Relationenschemas im Data Dictionary (ein neues Attribut wird dem Relationenschema Lehrstühle zugeordnet)
- ◆ Erweiterung der existierenden Basisrelation um ein Attribut, das bei jedem existierenden Tupel mit **null** besetzt wird

alter- und drop-Klausel für Attribute

- Die Klausel

```
alter spaltenname default_änderung
```

nur Änderung der Defaultwerte, nicht Änderung von Datentypen

- Die Klausel

```
drop spaltenname { restrict | cascade }
```

erlaubt Löschen von Attributen, falls

- ◆ keine Sichten und Integritätsbedingungen mit Hilfe dieses Attributs definiert wurden (im Fall **restrict**)
- ◆ oder mit gleichzeitiger Löschung dieser Sichten und Integritätsbedingungen (im Fall **cascade**)

Schlüsselbedingung simuliert mittels Index-Definition

```
create table Bücher (
  ISBN char(10) not null,
  Titel varchar(200),
  Verlagsname varchar(30))
```

```
create unique index Buchindex
  on Bücher
  (ISBN asc)
```

Beispiel in CODASYL-DDL

```
record Personen
  1 PANr pic 9999
  1 Name
    2 Vorname pic x(20)
    2 Nachname pic x(40)
  1 Adresse
    2 PLZ pic 9999
    2 Ort pic x(30)
    2 Straße pic x(60)
    ...
...
set Ausleihe
owner is Personen
member is Bücher
```

Die Anweisung create index

SQL-89: Bestandteil der Norm

```
create [unique] index indexname
  on basisrelationenname
  (spaltenname_1 ordnung_1,
   ...,
   spaltenname_k ordnung_k)
```

Netzwerkmodell: CODASYL-DDL

record	<i>R</i>		set	<i>S</i>
1	feld1	typ1	owner	is <i>O</i>
1	feld2		member	is <i>M</i>
	2	feld21		
	2	feld22		
	...			

Record- und Set-Typen in CODASYL-DDL (1971)

Hierarchisches Modell: IMS-DDL II

- Record-Informationen
 - ◆ Felder (analog der CODASYL-DDL; hier in Beispielen jedoch SQL-Datentypen)
 - ◆ die Position des Record-Typs im Baum:
 - **root** (Record-Typ ist Wurzel des Baumes) oder
 - **parent** = name_des_Vorfahren_im_Baum oder
 - ◆ virtuelle Felder:
 - virtual record_Typ in** name_des_Baumes

Beispiel in IMS-DDL II

```
tree Vorlesungen_Baum:  
  record Vorlesungen: root  
    V_Bezeichnung varchar(80)  
    SWS smallint  
    Semester smallint  
    Studiengang varchar(20)  
  record Hörer: parent = Vorlesungen  
    virtual Studenten in Studenten_Baum  
    Semester smallint
```

Hierarchisches Modell: IMS-DDL

- Bäume
 - tree** *B*: liste_von_Record-Typen
- Record-Typen
 - record** *R*: liste_von_Record-Informationen

Beispiel in IMS-DDL

```
tree Studenten_Baum:  
  record Studenten: root  
    PANr integer  
    Matrikelnummer varchar(10)  
    Studienfach varchar(20)  
    Immatrikulationsdatum date  
  record Buch_Exemplare: parent = Studenten  
    Inventarnr integer  
    ISBN char(10)  
    Auflage smallint  
  record Hörer: parent = Studenten  
    virtual Vorlesungen in Vorlesungen_Baum
```

Objektorientiertes Modell: ODL

```
interface Student : Person ( extent Studenten, key matrnr)
  attribute char matrnr[6];
  attribute string studienfach;
  attribute set<struct<float note, string fach>> zeugnis;

  relationship Person mutter inverse Person::kind;
  relationship Person vater inverse Person::kind;

float durchschnittsnote ()
  raises (keine_note);
void exmatrikulation (in string art)
  raises (buecher_ausgeliehen);
}
```

Erläuterung ODL-Beispiel II

- *Beziehungen* zu anderen Klassen mit dem Wortsymbol **relationship** — auch inverse Beziehungen: ermöglichen Wahl zwischen 1:1-, 1:n, und n:m-Kardinalitäten (hier: zwei 1:n-Beziehungen Vater und Mutter zwischen Studenten und Personen, da nur die Rückrichtung einen Set-Typ enthält: set<Person> kind)
- *Methoden* mit ihrer Schnittstelle und einer spezifizierten Ausnahmebehandlung, die im Fehlerfall ausgelöst wird, etwa bei Verletzung von Integritätsbedingungen

Original-IMS-DDL

Data Base Descriptions, Segments und Fields

statt Bäume, Record-Typen und Felder

```
dbd   name = STUDENTENBAUM
segm  name = STUDENTEN, BYTES = 40
field name = PANR, BYTES = 4, START = 1
field name = MATRIKELNUMMER, BYTES = 10, START = 5
field name = STUDIENFACH, BYTES = 20, START = 15
field name = IMMATRIKULATIONSdatum, BYTES = 6, START = 35
segm  name = BUCHEXEMPLARE, parent = STUDENTEN, BYTES = 16
field name = INVENTARNR, ...
...
```

Erläuterung ODL-Beispiel

Schnittstelle zum Objekttyp Personen beschreibt

- *Typhierarchie*: Angabe der Obertypen hinter dem Typnamen (hier: Obertyp Person)
- *Extension*, in der die aktuell erzeugten Objekte vom Typ Student gesammelt werden sollen (hier: Extension mit dem Namen Studenten)
- *Schlüssel* des Objekttyps, eine Auswahl der Attribute, die zur eindeutigen Identifizierung der Objekte unabhängig von der Objektidentität verwendet werden können (hier: nur das Attribut matrnr)
- *Attribute* mit Datentypen und Namen