2. Datenbankdefinition

$\leadsto$ **SQL-DDL** (Data Definition Language):

Teil der Standardsprache SQL für relationale Datenbanksysteme

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

- Typen
- Wertebereichen
- Relationenschemata
- Integritätsbedingungen

zu tun haben

- 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

SQL als Definitionssprache

- *Externe Ebene*
  - **create view**
  - **drop view**
- *Konzeptuelle Ebene*
  - **create table**
  - **alter table**
  - **drop table**
- *Konzeptuelle Ebene (zusätzlich ab SQL-92)*
  - **create domain**
  - **alter domain**
  - **drop domain**
- *Interne Ebene*
  - **create index**
  - **alter index**
  - **drop index**

Anmerkung:
Die Sprache SQL wurde in verschiedenen Schritten, zunächst als SQL-89, SQL-92 und zuletzt als SQL:1999, standardisiert, wobei der jeweils vorangegangene Standard immer mehr erweitert wurde. Dies wird sich auch weiterhin fortsetzen.

Die Anweisung create table

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

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

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-89 Level 2 mit IEF:

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

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));
```

create table in SQL-92:

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

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

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

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

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

```
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 );
```

```
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)));
```


Die Anweisungen alter table, drop table

- Syntax des **alter table**-Kommandos (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

In SQL-92 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);
```

alter- und drop-Klausel für Attribute

- Die Klausel

  alter `spaltenname default_änderung`

  erlaubt 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**)

Die Anweisung **drop table**:

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

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

Die Anweisung create index

SQL-89: Bestandteil der Norm

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

Schlüsselbedingung simuliert mittels Index-Definition und **not null**:

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

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