Kapitel 8: Datenintegrität

Datenintegrität

Statische Bedingung (jeder Zustand)
Dynamische Bedingung (bei Zustandsänderung)

Bisher:

- Definition eines Schlüssels
- 1:N - Beziehung
- Angabe einer Domäne

Jetzt:

- Check-Klauseln
- Referentielle Integrität
- Trigger

Check-Klauseln

```
create table Professoren (
  Name varchar(20) not null,
  Rang char(2)     check (Rang in ('C2','C3','C4')),
  Raum int  unique check (Raum between 100 and 200)
)
```

Referentielle Integrität

R, S zwei Relationen mit Schemata **R,S**

κ Primärschlüssel von **R**

$\alpha \subset$ **S** heißt Fremdschlüssel :

- s.α nur null oder nur ungleich null
- s.α ungleich null $\Rightarrow \exists$ r $\in$ *R* mit r.κ = s.α

Erlaubte Änderungen

Einfügen in S $\Rightarrow$ Fremdschlüssel verweist auf Tupel in *R*

Ändern in *S* $\Rightarrow$ neuer Fremdschlüssel verweist auf Tupel in *R*

Ändern des Primärschlüssels in *R* $\Rightarrow$
kein Fremdschlüssel aus *S* hatte auf ihn verwiesen

Löschen des Primärschlüssels in *R* $\Rightarrow$
kein Fremdschlüssel aus *S* hatte auf ihn verwiesen

Referentielle Integrität in SQL

`unique` Schlüsselkandidat

`primary key` Schlüssel (not null)

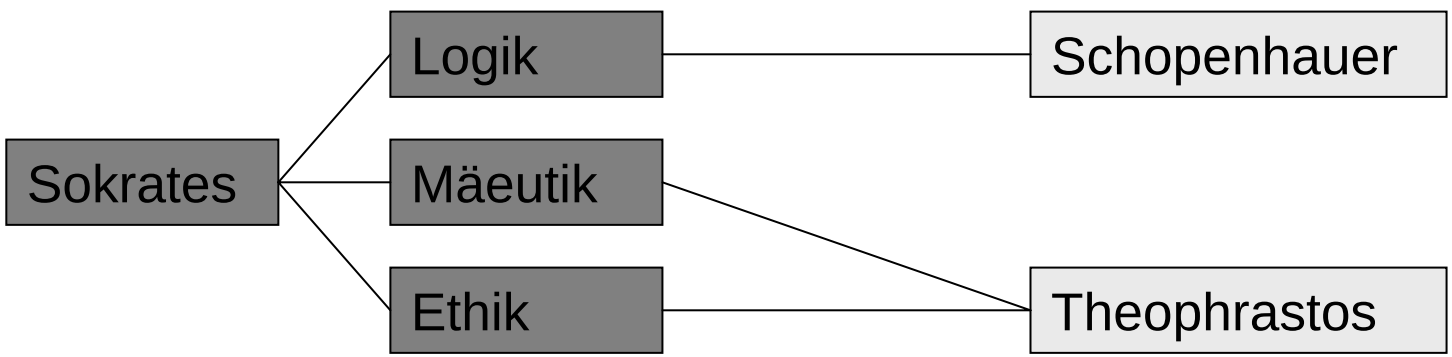
`foreign key` Fremdschlüssel (kann auch null sein)

```
boss int references Professoren
on update cascade
on delete cascade
on update set null
on delete set null
```

on delete cascade

```
create table hoeren (
  vorlnr  int references Vorlesungen
  on delete cascade
      ...
)

create table Vorlesung (
  gelesenvon int references Professoren
  on delete cascade
      ...
)
```



Referentielle Integrität im Uni-Schema

Professor darf nicht geändert oder entfernt werden:

```
references Professoren
```

Vorlesung darf geändert werden,

Vorlesung darf nicht entfernt werden:

```
references Vorlesung on update cascade
```

Student darf geändert und entfernt werden:

```
references Student on update cascade
                     on delete cascade
```

Studenten

```
CREATE TABLE Studenten(
   MatrNr      INTEGER PRIMARY KEY,
   Name        VARCHAR(20) NOT NULL,
   Semester    INTEGER,
   GebDatum    DATETIME
)
```

Professoren

```
CREATE TABLE Professoren(
   PersNr    INTEGER PRIMARY KEY,
   Name      VARCHAR(20) NOT NULL,
   Rang      CHAR(2) CHECK (Rang in ('C2','C3','C4')),
   Raum      INTEGER UNIQUE,
   Gebdatum  DATETIME
)
```

Assistenten

```
CREATE TABLE Assistenten (
        PersNr          INTEGER PRIMARY KEY,
        Name            VARCHAR(20) NOT NULL,
        Fachgebiet      VARCHAR(20),
        Boss            INTEGER REFERENCES Professoren,
        GebDatum        DATETIME
)
```

Vorlesungen

```
CREATE TABLE Vorlesungen (
        VorlNr          INTEGER PRIMARY KEY,
        Titel           VARCHAR(20),
        SWS             INTEGER,
        gelesenVon      INTEGER REFERENCES Professoren
)
```

hoeren

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

voraussetzen

```
CREATE TABLE voraussetzen (
 Vorgaenger INTEGER REFERENCES Vorlesungen
                                ON UPDATE CASCADE,
 Nachfolger INTEGER REFERENCES Vorlesungen
                                ON UPDATE CASCADE,
 PRIMARY KEY    (Vorgaenger, Nachfolger)
)
```

pruefen

```
CREATE TABLE pruefen (
  MatrNr INTEGER REFERENCES Studenten
                            ON UPDATE CASCADE
                            ON DELETE CASCADE,
  VorlNr INTEGER REFERENCES Vorlesungen
                            ON UPDATE CASCADE,
  PersNr INTEGER REFERENCES Professoren,
  Note   NUMERIC(3,1) CHECK (Note between 0.7 and 5.0),
  PRIMARY KEY    (MatrNr, VorlNr)
)
```

Trigger

Einer Tabelle zugeordnet:

after {update | delete | insert}

Einer Tabelle oder Sicht zugeordnet:

instead of {update | delete | insert}

Temporäre Tabellen deleted und inserted

Trigger korrigieredegradierung

```
create trigger korrigieredegradierung
on Professoren after update as
if update(Rang)
begin
  update professoren
  set rang = d.rang
  from professoren p, deleted d
  where p.persnr = d.persnr
  and p.rang < d.rang
end
```

Trigger verhinderedegradierung

```
create trigger verhinderedegradierung
on Professoren instead of update
as
  update professoren
  set rang = i.rang
  from deleted d, inserted i
  where professoren.persnr = d.persnr
  and  d.persnr = i.persnr
  and (d.rang=null or d.rang < i.rang)
```

Trigger befoerderung

```
create trigger befoerderung
on hoeren after insert, update
as
update professoren set rang='C4'
where persnr in
  (select persnr
   from   professoren p,
          vorlesungen v,
          hoeren h
   where  p.persnr = v.gelesenvon
   and    v.vorlnr = h.vorlnr
   group by p.persnr
   having count(*) > 10)
```

View Geburtstagsliste

```
create view Geburtstagsliste
as
select Name,
     datediff(year,gebdatum,getdate()) as Jahre
from Person
```

Trigger Geburtstag

```
create trigger Geburtstag
on Geburtstagsliste
instead of insert
as
insert into Person (name,gebdatum)
select i.name,
       dateadd(year, -i.jahre, getdate())
from inserted i
```