

# Datenbanksysteme 2011

## Kapitel 8: Datenintegrität

Oliver Vornberger

Institut für Informatik  
Universität Osnabrück

# 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  $\mathbf{R}, \mathbf{S}$

$\kappa$  Primärschlüssel von  $\mathbf{R}$

$\alpha \subset \mathbf{S}$  heißt Fremdschlüssel :

- $s.\alpha$  enthält nur Nullwerte oder nur Werte ungleich null
- $s.\alpha$  ungleich null  $\Rightarrow \exists r \in R$  mit  $r.\kappa = s.\alpha$

# 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)

**FOREIGN KEY** gelesenvon **REFERENCES** Professoren(persnr)

**ON UPDATE CASCADE**

**ON DELETE CASCADE**

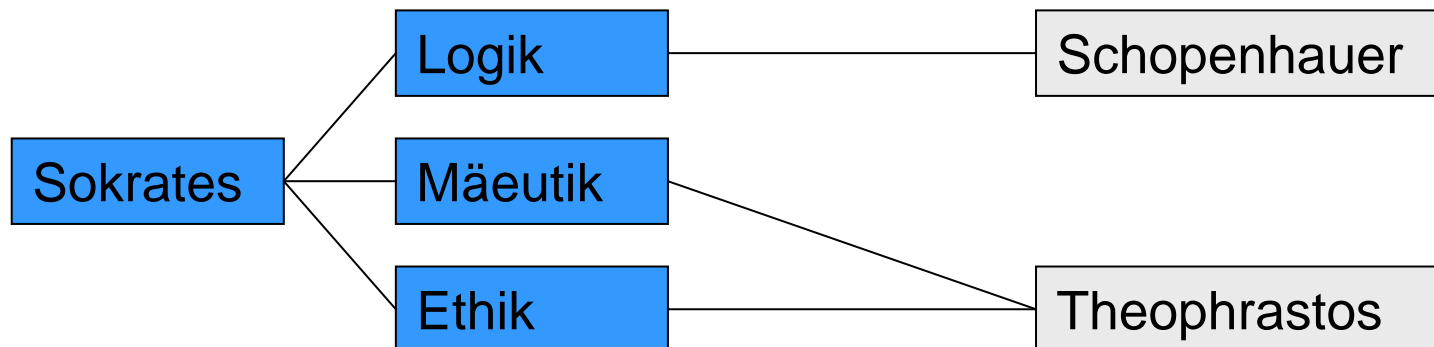
**ON UPDATE SET NULL**

**ON DELETE SET NULL**

# on delete cascade

```
CREATE TABLE hoeren (  
  vorlnr int,  
  FOREIGN KEY vorlnr REFERENCES Vorlesungen(vorlnr)  
  ON DELETE CASCADE  
  ...  
)
```

```
CREATE table Vorlesung (  
  gelesenvon int,  
  FOREIGN KEY gelesenvon references Professoren(persnr)  
  on DELETE CASCADE  
  ...  
)
```



# Referentielle Integrität im Uni-Schema

**REFERENCES Professoren**

Professor darf nicht geändert oder entfernt werden:

**REFERENCES Vorlesung ON UPDATE CASCADE**

Vorlesung darf geändert werden,

Vorlesung darf nicht entfernt werden:

**REFERENCES Student ON UPDATE CASCADE  
ON DELETE CASCADE**

Student darf geändert und entfernt werden:



# Studenten

```
CREATE TABLE Studenten(  
    matrnr      INTEGER PRIMARY KEY,  
    name        VARCHAR(20) NOT NULL,  
    semester    INTEGER,  
    gebdatum    DATE  
)
```

# 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  DATE  
)
```

# Assistenten

```
CREATE TABLE Assistenten (  
  persnr          INTEGER PRIMARY KEY,  
  name            VARCHAR(20) NOT NULL,  
  fachgebiet     VARCHAR(20),  
  boss           INTEGER,  
  gebdatum       DATE,  
  FOREIGN KEY(boss) REFERENCES Professoren(persnr)  
)
```

# Vorlesungen

```
CREATE TABLE Vorlesungen (  
  vorlnr          INTEGER PRIMARY KEY,  
  titel           VARCHAR(20),  
  sws             INTEGER,  
  gelesenVon      INTEGER,  
  FOREIGN KEY(gelesenvon) REFERENCES Professoren(persnr)  
)
```

# hoeren

```
CREATE TABLE hoeren (  
    matrnr INTEGER,  
    VorlNr INTEGER,  
  
    PRIMARY KEY(matrnr,vorlnr),  
  
    FOREIGN KEY(matrnr) REFERENCES Studenten(matnr)  
        ON UPDATE CASCADE  
        ON DELETE CASCADE,  
  
    FOREIGN KEY(vorlnr) REFERENCES Vorlesungen(vorlnr)  
        ON UPDATE CASCADE,  
  
)
```

# voraussetzen

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

# pruefen

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

# Trigger

Löst Aktionen bei bestimmten Bedingungen aus.

Einer Tabelle zugeordnet:

```
create trigger <name>  
{before | after}      {insert | update | delete }  
on <table>
```

Temporäre Tabellen OLD und NEW

im Insert-Trigger nur NEW

im Delete-Trigger nur OLD

im Update-Trigger NEW und OLD



# Attribut anzhoeerer

Speichere bei jedem Professor die Zahl seiner Hörer:

```
update Professoren p
set anzhoeerer = (select count(*)
                  from Vorlesungen v, hoeren h
                  where p.PersNr = v.gelesenVon
                  and h.VorlNr = v.VorlNr)
```

A

# Trigger zaehlehoerer

```
drop trigger if exists zaehlehoerer

create trigger zaehlehoerer
after insert on hoeren
for each row
begin
    update Professoren p
    set p.anzhoerer = p.anzhoerer + 1
    where NEW.vorlnr in
    (select vorlnr from Vorlesungen
     where gelesenvon = p.persnr);
end
```

# Trigger korrigieredegradierung

```
drop trigger if exists korrigieredegradierung  
  
create trigger korrigieredegradierung  
after update on Professoren  
for each row  
begin  
    UPDATE Professoren SET rang = OLD.rang  
    WHERE rang < OLD.rang;  
end
```

Nicht möglich, da in MySQL ein Trigger nicht die Tabelle ändern darf, die ihn ausgelöst hat

# Trigger befoerderung

```
drop trigger if exists befoerderung

create trigger befoerderung
after insert on hoeren
for each row
  update Professoren set rang='C4'
  where persnr in
    (select gelesenvon
     from hoeren h, Vorlesungen v
     where h.vorlnr = v.vorlnr
     group by gelesenvon
     having count(*) > 2)
```

# Table Protokoll

```
create table Protokoll (  
    zeitpunkt timestamp,  
    bemerkung text  
)
```

```
drop trigger if exists schreibprotokoll  
  
create trigger schreibprotokoll  
after insert on Professoren  
for each row  
    insert into Protokoll (bemerkung)  
    values (CONCAT(NEW.persnr, ' wurde eingefuegt.'))
```

# View Geburtstagsliste

```
create or replace view Geburtstagsliste  
as  
select name, year(NOW())- year(gebdatum) as Jahre  
from Personen
```

# Trigger Geburtstag

Wunsch:

```
insert into Geburtstagsliste('Thorben', 25)
```

```
create or replace trigger Geburtstag
instead of insert on Geburtstagsliste
for each row
  insert into Personen (name, gebdatum)
  values (NEW.name,
         DATE_ADD(NOW(), interval -NEW.Jahre year)
```

Aber: in MySQL kein Trigger für Views, kein INSTEAD OF

# Table Geburtstagstabelle

```
create table Geburtstagstabelle (  
    name varchar(30),  
    jahre int  
)
```

```
drop trigger if exists Geburtstageintragen  
  
create trigger Geburtstageintragen  
after insert on Geburtstagstabelle  
for each row  
    insert into Personen (name, gebdatum)  
    values(NEW.name,  
          DATE_ADD(NOW(),interval -NEW.jahre year))
```