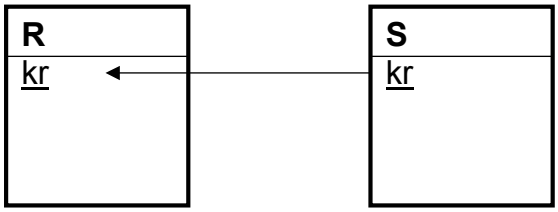
Kap 6.7 Integritätsbedingungen in SQL

Arten von Integritätsbedingungen

- Schlüssel und Eindeutigkeit
- Fremdschlüssel, referentielle Integrität
- Typen von Attributen, aber niedrige Semantik, z.B. Tel# = Zimmer# ??
- Bereiche von Attributwerten



sei $s \in S$, dann muß sein $s.kr$ = **null**
oder $s.kr \in \pi_{kr} R$

Diese Bedingung heißt ***referentielle Integrität*** und wird vom DBS automatisch überwacht.

$\pi_{kr} S \subseteq \pi_{kr} R =: KR$

Änderungs-Bedingungen:

1. **insert into** S **values** s — $s.kr \in \pi_{kr} R$
2. ändere s.kr zu s.kr' — $s.kr' \in \pi_{kr} R$
3. verändern von r.kr — $\neg\exists\, s \in S: s.kr = r.kr$
4. löschen von r mit r.kr — $\neg\exists\, s \in S: s.kr = r.kr$

Hinweis: durch Schlüssel-Fremdschlüssel Beziehungen entstehen automatisch Integritätsbedingungen!

Schlüssel-Vereinbarung

```
create table  R
    ( kr: integer primary key
    …)
```

Fremdschlüssel-Vereinbarung

```
create table S
    ( ...
      m: integer references R )
```

m bezieht sich automatisch auf ***primary key*** von R

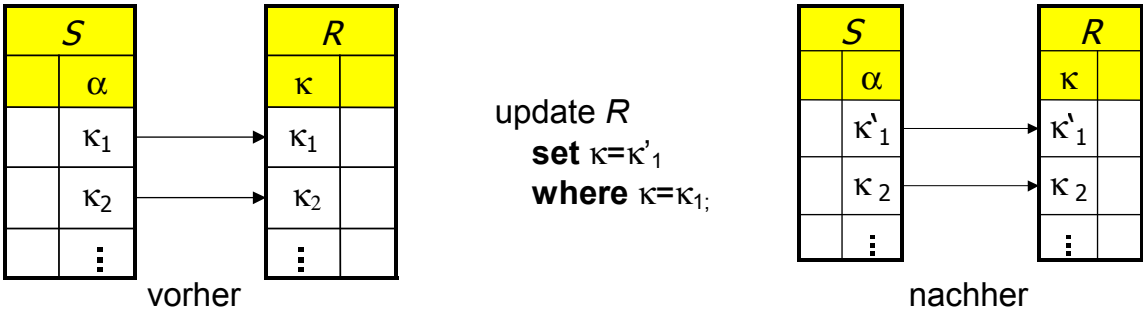
Beispiel

```
create table Student
    ( Matr#      integer primary key,
      Name       varchar (25),
      Semester   tinyint )

create table prüft
    ( Prof       varchar (25),
      Matr#      integer references Student,
      Fach:      varchar (17) )
```

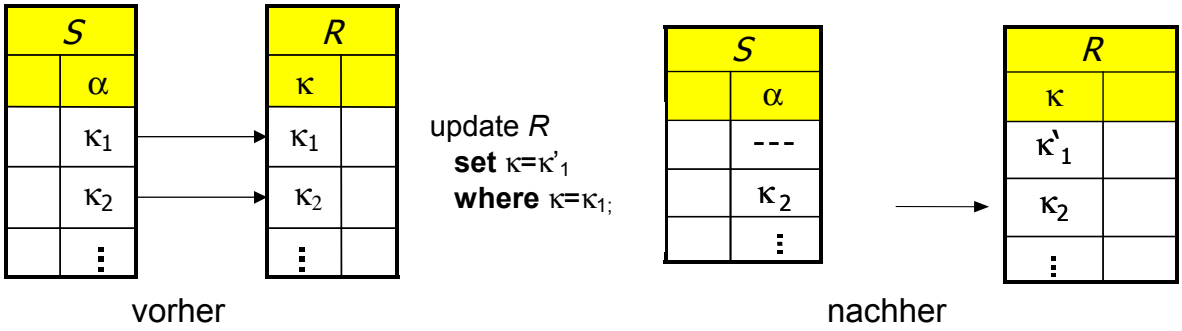
nach Kemper/Eickler: Kaskadierung

create table *S* (..., α **integer references** *R* **on update cascade**);

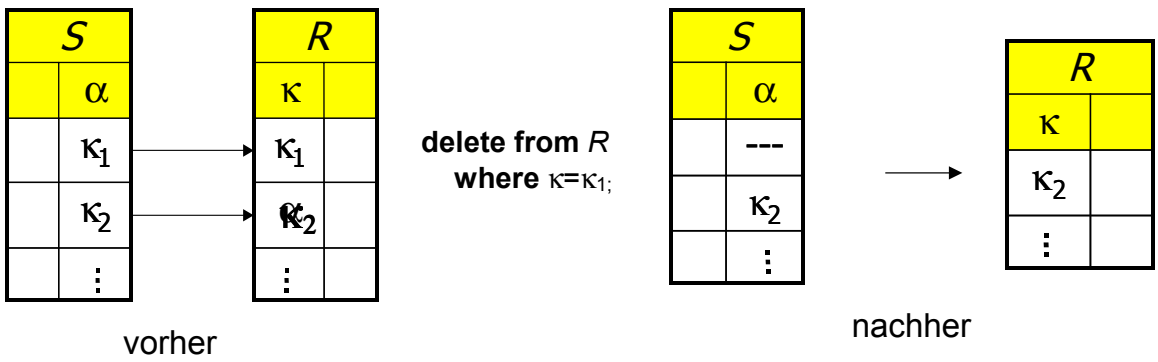


create table *S* (...,α **integer references** *R* **on delete cascade**);

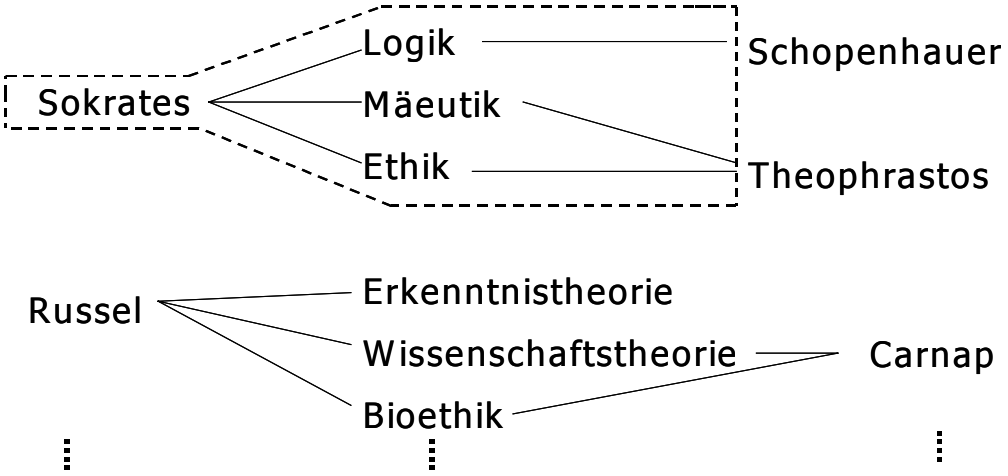
create table *S* (..., α **integer references** *R* **on update set null**);



create table *S* (...,α **integer references** *R* **on delete set null**);



delete from Professoren **where** Name = ‚Sokrates';



create table Vorlesungen
(...,
gelesenVon **integer**
references Professoren
on delete cascade);

create table hören
(...,
VorlNr **integer**
references Vorlesungen
on delete cascade);

create table Studenten
(MatrNr **integer primary key**,
Name **varchar**(30) **not null**,
Semester **integer check** Semester **between** 1 **and** 13),

create table Professoren
(PersNr **integer primary key**,
Name **varchar**(30) **not null**,
Rang **character**(2) **check** (Rang **in** (`C2´,`C3´,`C4´)),
Raum **integer unique**);

```
create table Assistenten
    ( PersNr        integer primary key,
    Name            varchar(30) not null,
    Fachgebiet      varchar(30),
    Boss            integer,
    foreign key     (Boss) references Professoren on delete set
                    null);

create table Vorlesungen
    ( VorlNr        integer primary key,
    Titel           varchar(30),
    SWS             integer,
    gelesen Von     integer references Professoren on delete set
                    null);
```



```
create table hören
    ( MatrNr        integer references Studenten on delete
                    cascade,
    VorlNr          integer references Vorlesungen on delete
                    cascade,
    primary key     (MatrNr, VorlNr));

create table voraussetzen
    ( Vorgänger     integer references Vorlesungen on delete
                    cascade,
    Nachfolger      integer references Vorlesungen on
                    delete cascade,
    primary key     (Vorgänger, Nachfolger));
```

```
create table prüfen
    ( MatrNr        integer references Studenten on delete
                    cascade,
    VorlNr          integer references Vorlesungen,
    PersNr          integer references Professoren on delete
                    set null,
    Note            numeric (2,1)  check (Note between 0.7
                    and 5.0),
    primary key     (MatrNr, VorlNr));
```



```
create table prüfen
    ( MatrNr        integer references Studenten on delete
                    cascade,
    VorlNr          integer references Vorlesungen,
    PersNr          integer references Professoren on delete
                    set null,
    Note            numeric (2,1)  check (Note between 0.7
                    and 5.0),
    primary key     (MatrNr, VorlNr));
    constraint      VorherHören
        check (exists( select *
                       from hören h
                       where  h.VorlNr = prüfen.VorlNr and
                              h.Matr.Nr = prüfen.MatrNr))
);
```