Why use a DBMS? (Week 1)

- Avoid redundancy and **inconsistency**
- Rich (declarative) access to the data
- Synchronize concurrent data access
- Recovery after system failures
- Security and privacy

- Reduce cost and pain to do something useful
  - There is always an alternative!!!

Integrity of Data

- Example Constraints
  - Keys
  - multiplicity of relationships
  - attribute domains
  - subset relationship for generalization
  - Referential integrity (foreign keys -> keys)

- Static Constraints
  - Constraints that any instance of a DB must meet

- Dynamic Constraints
  - Constraints on a state transition of the DB

Who checks? DB vs. App

- Why implement constraints in the DB?
  - Good way to annotate & document schema
  - DB is a central point (once and for all cases)
  - Safety net: in case you forget it in the app
  - Useful for DB-level optimization
    - Constraint: all students are older than 18 years.
    - Query: SELECT * FROM Student WHERE age < 17;
    - Query can be evaluated without looking at any student.

- Why implement constraints in the App?
  - Meaningful error messages.

- **It is important to do both!!!**

Referential Integritity Constraints

Foreign Keys

- Refer to tuple from a different relation
- E.g., PersNr in Lecture refers to a Professor

Definition: Referential Integritity

- For every foreign key one of the two conditions must hold
  - the value of the foreign key is *NULL* or
  - the referenced tuple must exist

- (Example on the Web: 404 Error becomes impossible)

Referential Integritity in SQL

- SQL Syntax to declare keys and foreign keys:
  - Key: **unique**
  - Primary key: **primary key**
  - Foreign key: **foreign key**

- Example:

create table *R*
(α **integer primary key**,
β **varchar(30) unique,**
...);

create table *S*
(...,
κ **integer references** *R*);

Maintaining referential integrity?

Updates of referenced data which result in a violation

1. Default: reject the update (return an error)
2. **cascade:** propagate update
3. **set null:** set references to null

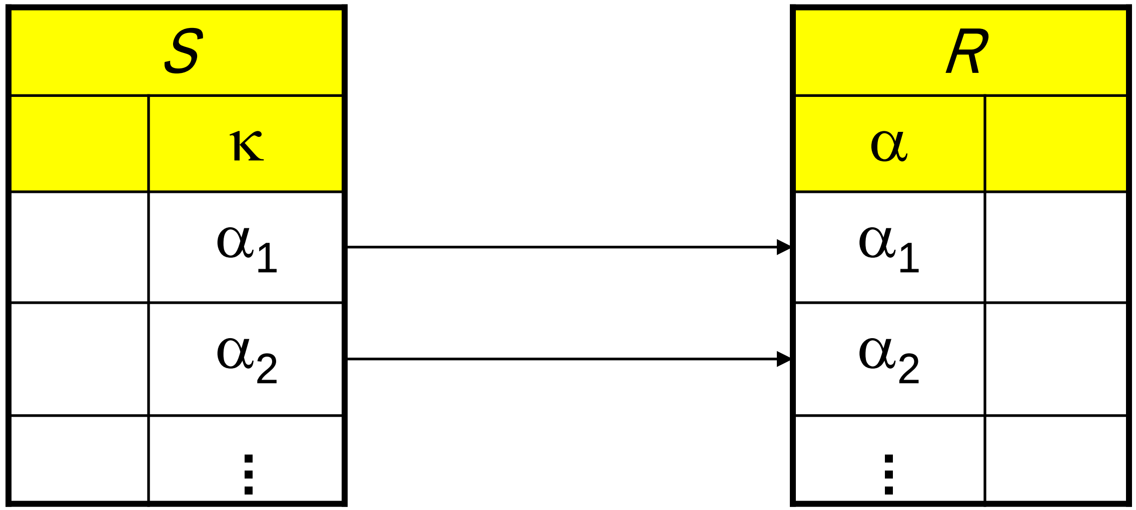
4. (Set references to default value. Not supported in SQL.)

The right choice depends on the ER model

- e.g. weak vs. strong entities
- relations that implement N:M relationships
- 1:N relations
- Exercise: extend rules for ER->relational translation!

Maintaining referential integrity

Original



Update

update *R*

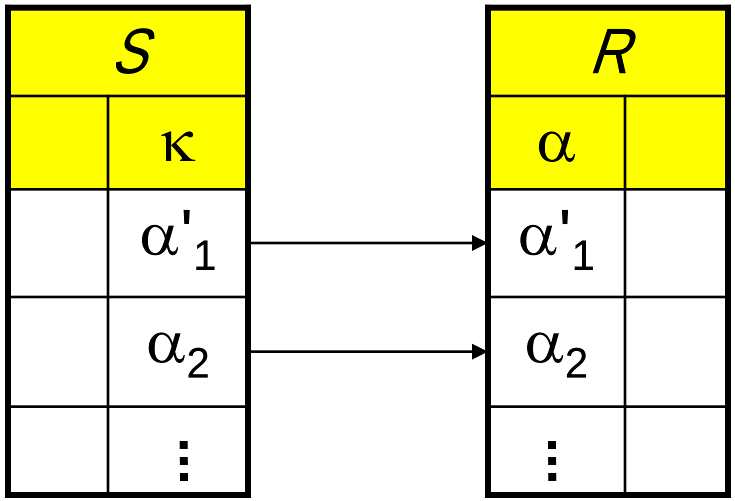
set α = α'_1

where α = α_1;

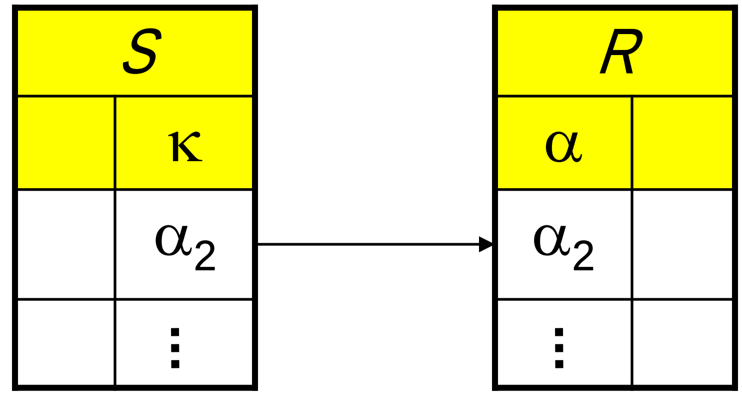
delete from *R*

where α = α_1;

Cascade (weak entities, n:m relationships)



Update of S



Delete in S

create table *S*

(…,

κ **integer references** *R*

on update cascade);

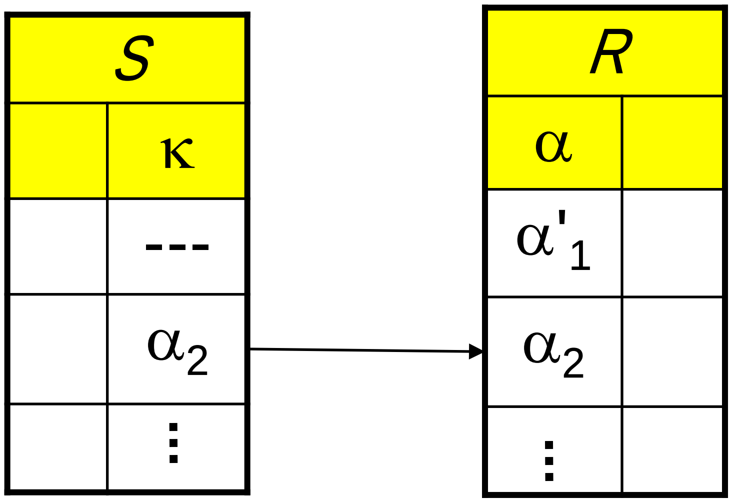
create table *S*

(…,

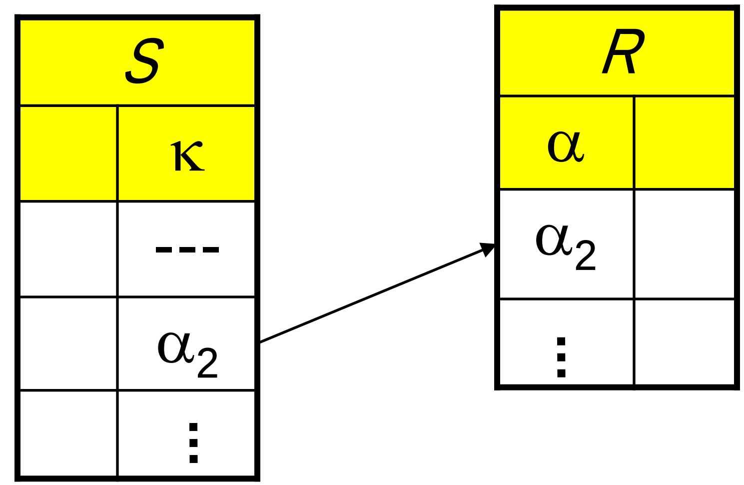
κ **integer references** *R*

on delete cascade);

Set Null (strong entities)



Update of S



Update of S

create table *S*

(…,

κ **integer references** *R*

on update set null);

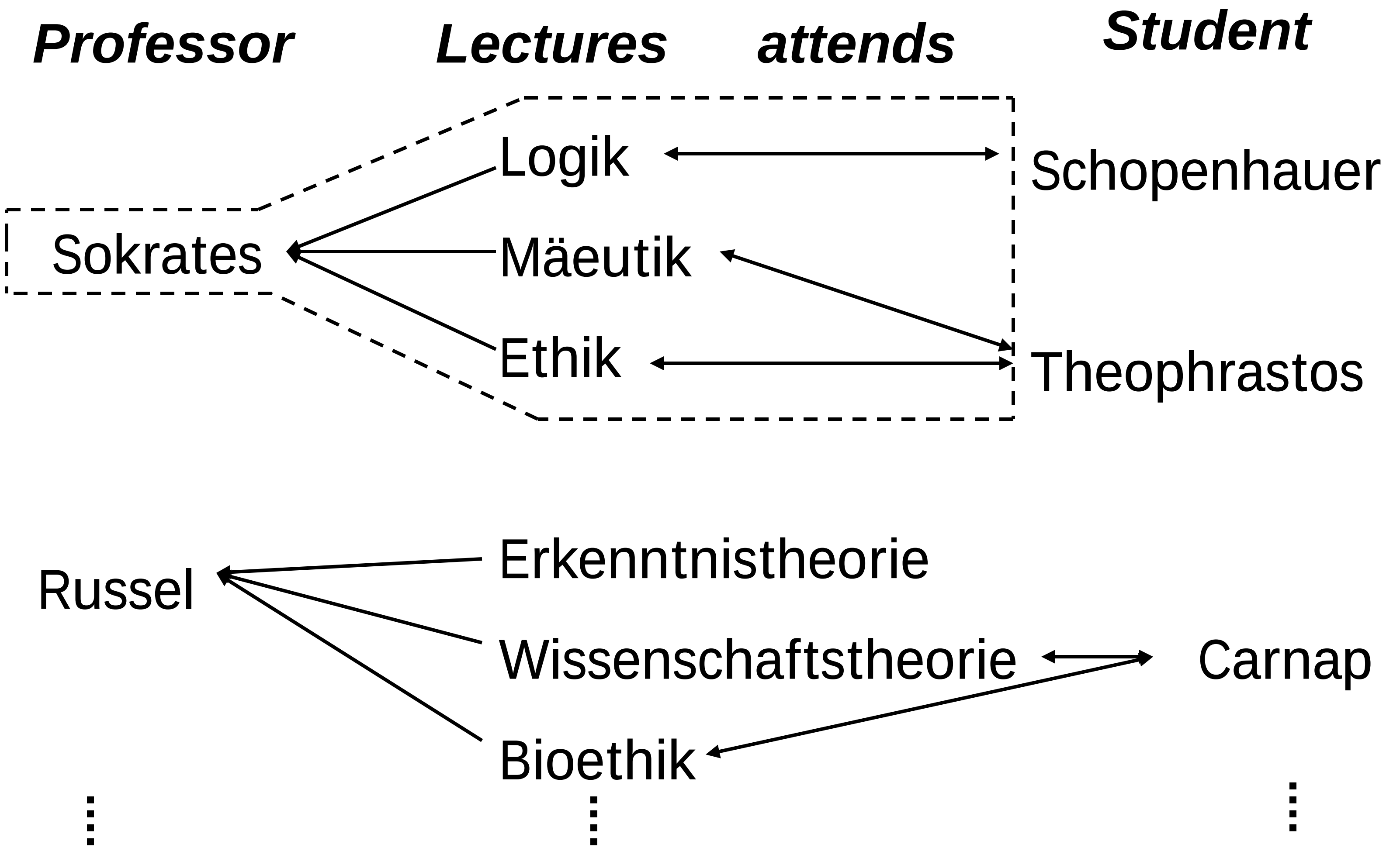
create table *S*

(…,

κ **integer references** *R*

on delete set null);

Cascading Deletes



```
create table Lecture
    ( ...,
    PersNr   integer
                    references Professor
                    on delete cascade);

create table attends
      ( ...,
      Nr   integer
               references Lecture
               on delete cascade);
```

Constraints on Domains

- Integer domains

  … **check** Semester **between** 1 **and** 13

- Enum types

  … **check** Level **in** (`Assistant´, `Associate´, `Full´) …

Uni-DB schema with Constraints

create table Student

(Legi **integer primary key**,

Name **varchar**(30) **not null**,

Semester **integer check** Semester **between** 1 **and** 13),

create table Professor

(PersNr **integer primary key**,

Name **varchar**(30) **not null**,

Level **character**(2) **check** (Level **in** (`AP´,`CP´,`FP´)),

Room **integer unique**);

```
create table Assistant
    ( PersNr        integer primary key,
    Name            varchar(30) not null,
    Area            varchar(30),
    Boss            integer,
    foreign key     (Boss) references Professor
                    on delete set null);
create table Lecture
    ( Nr            integer primary key,
    Title           varchar(30),
    CP              integer,
    PersNr          integer references Professor
                    on delete set null);
```

```
create table attends
  ( Legi          integer references Student
                  on delete cascade,
    Nr            integer references Lecture
                  on delete cascade,
    primary key   (Legi, Nr));
create table requires
  ( Prerequisite  integer references Lecture
                  on delete cascade,
    Follow-up     integer references Lecture
                  on delete cascade,
    primary key   (Prerequisite, Follow-up));
```

```
create table tests
    ( Legi          integer references Student
                    on delete cascade,
    Nr              integer references Lecture,
    PersNr          integer references Professor
                    on delete set null,
    Grade           numeric (3,2)
                    check (Grade between 1.0 and 6.0),
    primary key     (Legi, Nr));
```

1:1 Relationships (Wedding)

```
create table Man(
   name   varchar(30)  primary key;
   spouse varchar(30)  references Woman);
create table Woman(
   name    varchar(30) primary key;
   spouse  varchar(30) references Man);
```

- Legal: Helga marries Hugo, but Hugo does not marry Helga.
  - Mutual marriage cannot be expressed in SQL.
  - How would you model marriage in SQL?

- N.B.: The real implementation is based on **transactions!**

Trigger (ECA Rules)

```
create trigger noDegradation
before update on Professor
for each row
when (old.Level is not null)
begin
    if :old.Level = ‘Associate' and :new.Level = ‘Assistant' then
        :new.Level := ‘Associate';
    end if;
    if :old.Level = ‘Full' then
        :new.Level := ‘Full'
    end if;
    if :new.Level is null then
      :new.Level := :old.Level;
    end if;
end
```

Dangers of Triggers

```
create trigger weddingMan
after update on Man
for each row
when (true)
begin
    update Woman set spouse = :new.Name
    where name = :new.spouse;
    update Woman set spouse = null
    where name = :old.spouse;
end
```

- What happens if we write a weddingWoman trigger?
- Is marriage better modeled statically or dynamically?