Systems@ETH zürich

Data Modeling and Databases
Ch 5: Integrity Constraints

Gustavo Alonso
Systems Group
Department of Computer Science
ETH Zürich

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Controling the data

- A schema defines the domain of the data stored in the database and what are the concepts to be captured (entities and relations)
- Types determine the format and space reserved for the values that attributes can take (also determine how the data will be processed, but we will not look into this here)
- We are still missing some more tools to control that the data in the database is correct.

Constraints

- Think about pre- and post-conditions or contracts in programming languages.
- Constraints are the way a database makes sure changes are consistent and do not cause trouble later on.
- Constraints control the content of the data and its consistency as part of the schema
- Transactions (concurrency control and recovery) control the data when concurrent access and failures occur -> more about this later in the course

Problems that occur …

- Inserting tuples without a key (never to be found again)
- Adding references to tuples that do not exist (e.g., lecture taught by unknown professor, test taken by unknown student)
- Nonsensical values for attributes (negative age, negative salary,)
- Conflicting tuples (e.g., lecture with two entries, each one with different credits)
- …

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 Integrity 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 );
```

From the manuals

- A FOREIGN KEY constraint can reference columns in tables in the same database or within the same table. These are called self-referencing tables. For example, consider an employee table that contains three columns: **employee_number**, **employee_name**, and **manager_employee_number**. Because the manager is also an employee, there is a foreign key relationship from the **manager_employee_number** column to the **employee_number** column. (Microsoft SQL Server)

- InnoDB requires indexes on foreign keys and referenced keys so that foreign key checks can be fast and not require a table scan. In the referencing table, there must be an index where the foreign key columns are listed as the *first* columns in the same order. (MySQL)

Maintaining referential integrity

Actions in SQL:2003

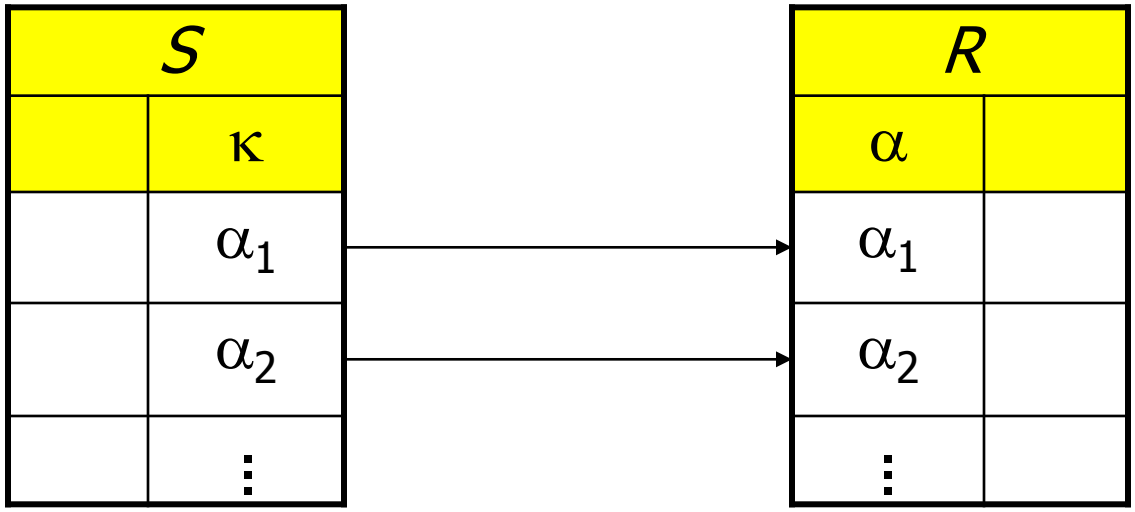
- **cascade:** propagate update or delete
- **restrict:** prevents deletion of the primary key ("master" table) before trying to do the change, causes an error
- **no action:** prevents modifications after attempting the change (but triggers might be executed), causes an error
- **set default, set null:** set references to null or to a default value

How does this work? Triggers

- The database can trigger certain Actions given a Condition when an Event occurs
  - Event
  - Condition
  - Action
  - ECA rule
- Integrity constraints are often implemented as system triggers
- Users can also crate triggers

Maintaining referential integrity

Original



Update

update *R*

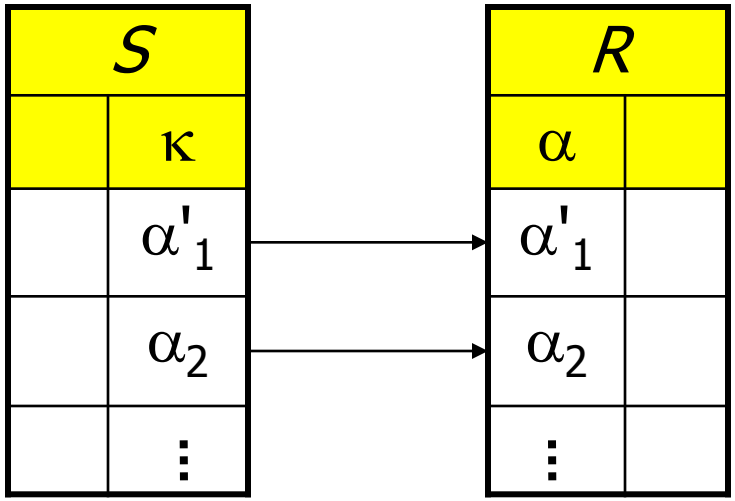
set $\alpha = \alpha'_1$

where $\alpha = \alpha_1$;

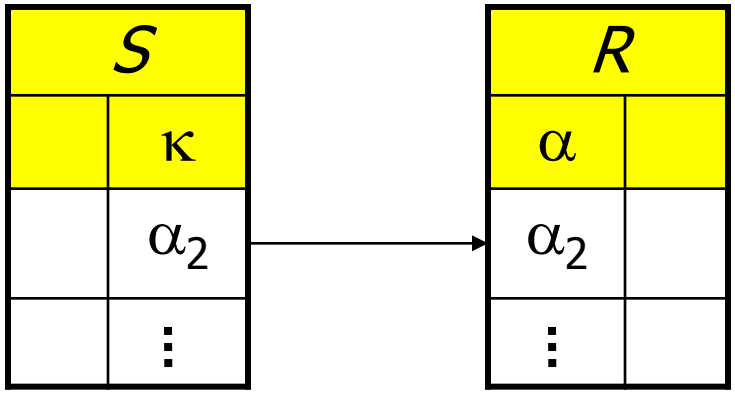
delete from *R*

where $\alpha = \alpha_1$;

Cascade (weak entities, n:m relationships)



Update of S



Delete in S

create table *S*

(...,

κ **integer references** *R*

on update cascade);

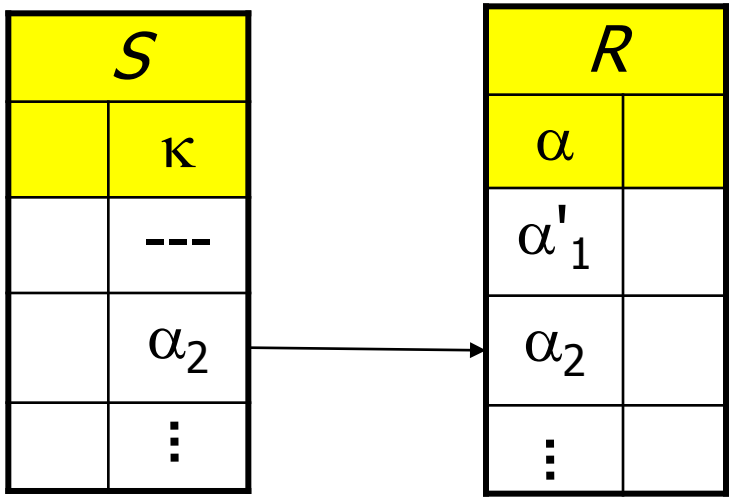
create table *S*

(...,

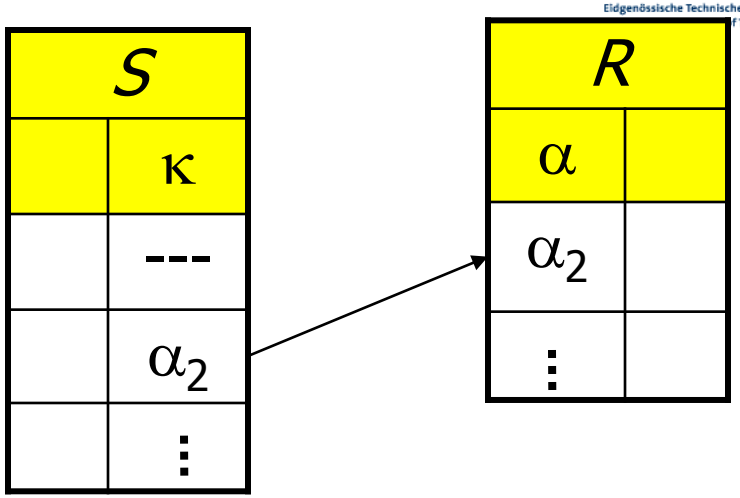
κ **integer references** *R*

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 updates/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´) ...

Constraints across attributes

- Example with Oracle notation

```
create table contract
   (begin_date number,
    end_date number,
    salary  number,
    check (begin_date < end_date)
);
```

Uni schema with Constraints

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

```
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 );
```

Uni schema with Constraints

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

```
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);
```

Uni schema with Constraints

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

```
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));
```

Uni schema with Constraints


```
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);
```

- Schema allows the following: X marries Y, but Y does not marry X.
- Mutually exclusive relations need additional constraints
- How would you model marriage in SQL?

Marriage

```
CREATE TABLE People
   (person_ID VARCHAR(20) NOT NULL UNIQUE);

CREATE TABLE Marriages
   (spouse_1 NOT NULL UNIQUE REFERENCES
   People (person_ID),
     spouse_2 NOT NULL UNIQUE REFERENCES
   People (person_ID),
   CHECK (spouse_1 < spouse_2) );
```

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?