relational model

excerpt of the real world

conceptual schema

relational schema / relational model

XML schema

network schema

object-oriented schema
relational model - properties

• set oriented – different to network or hierarchical models
• simple structure – everything is stored in tables (called relations)
• rows of the table are tuples and columns are attributes
• primary key is underlined
transformation of conceptual schema into relations

• E / R model has two fundamental structures
  – entities
  – relations
• relational model has ONLY RELATIONS (tables)

transformation of conceptual to relational is a two – step process
relational model

1. Initial transformation
   • entities are transferred easily: all attributes of the entity become an attribute of the relation
   • relations (from E/R) get the key attributes of the connecting entities and their own

   \[
   \text{Students: } \{[\text{studnr: integer, name: string, semester: integer}]\}
   \]

   \[
   \text{attend (N:M): } \{[\text{studnr: integer, lecturenr: integer}]\}
   \]
   • for (1:N) or (N:1) relations only the key attribute of the N part becomes key
   • give (1:N): \{[\text{PersNr: integer, lecturenr: integer}]\}

2. Refinement for (1:N) or (N:1) and (1:1)
   • relations with the same primary key can be condensed
relational model – example

**trainstations**:

- **name**: string
- **#platforms**: integer

**towns**:

- **name**: string
- **federal state**: string

**Is_in**:

- **name**: string
- **federal state**: string
- **name**: string

(1:N) relation: one train station is in only one city.

**trainstation**: {**name**, **#platforms**, **is_in**}

**towns**: {**name**, **federal state**}
DDL: Data Definition Language

Part of SQL that is used for Data Definition to:
- Define the schema
- Control access to the DB

Typical statements:
- Create/drop tables
- Create/drop views
- Create/drop indexes

Supported data types: numbers, strings, dates etc.
Integrity Constraints

Guarantees the consistency of the data.

Typical integrity constraints:
- Primary key constraint
- NOT NULL
- UNIQUE
- DEFAULT
- CHECK clauses
Referential Integrity

- Foreign key constraint
- Constraint that links tables with each other
- Prevent actions that destroy that link
- Variants: SET NULL, CASCADE

```sql
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID)
    REFERENCES Persons(PersonID)
);

CREATE TABLE Orders (
    OrderID int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID) ON DELETE SET NULL
);
```
Example exercises

- Provide SQL for:
  - New table **lectures** with constraint that weekly hours are between 2 and 6
  - New table **students** with semester set to 1 by default
- The following code is given:

```sql
CREATE TABLE Assistants (PersNr int NOT NULL PRIMARY KEY, Name varchar(255) NOT NULL, Area varchar(255), Boss int, FOREIGN KEY (Boss) REFERENCES Professors(PersNr) ON DELETE CASCADE);

DELETE FROM Professors WHERE Name = 'Kopernikus';

SELECT Name FROM Assistants;
```