2. Conceptual Database Design

2.1 Phases of Database Design

Data oriented approach

- Which data have to be managed in the system?
- How are the data changed in the system?

Stages of database design
Requirements analysis

- based on the knowledge about information structure requirements of the interesting part of the real world, e.g.
  - What are the relevant objects and their attributes?
  - What are the relationships between the objects?
  and about requirements concerning data processing, e.g.
  - What are the typical operations?
  - meaning of operations, runtime of operations, amount of data

- central problem of requirements analysis:
  - user and application specialist, respectively, must transfer information to the developer, i.e., communication between the developer and the user is necessary (problem of different terminologies)
  - no general approach for a successful requirements analysis available (software engineering problem)
Conceptual design

- specification of data structures in a formal language on the basis of a **conceptual data model** of high abstraction

- most known conceptual data model is the **Entity-Relationship Model** (E-R Model)

- only schema information but not instances (data) are considered
  ⇒ only a DDL but not a DML necessary

- transformation of a requirements specification into a conceptual design is difficult:
  - users employ different terms/names for the same object type
  - users employ the same term for different object types
  - omitting irrelevant structures (abstraction of real objects)
Logical design

- mapping of the data structures of the conceptual model into data structures of a **logical** (implementation) **data model**, i.e., in concrete data types of an underlying database model
- data types of the logical model abstract from their physical representation
- goal: unique storage of data (avoidance of redundancy)

Physical design

- mapping of the data structures of the logical design to files (**physical data model**)
- balanced use of index structures to support queries efficiently
  - too many indexes: updates of the database become too expensive
  - too few indexes: retrieval operations are not supported efficiently
2.2 The Entity-Relationship Data Model

Introduction

- most known conceptual data model on a high abstraction level, easy to understand, independent of aspects of data organization and data management
- E-R model has great importance in practice
- two-phase procedure for DB design
  - phase 1: requirements analysis and design of an E-R model
  - phase 2: transformation of the E-R model into a concrete logical model
- goal: modeling of an interesting part of the “real world” by **abstraction** so that questions about it can be answered with the aid of the model

E-R model describes the “real world” by

- **entities** (objects)
- **attributes** (properties)
- **relationships** between entities
Entity

- **entities** are distinguishable, physically or intellectually existing concepts of the mini-world to be modeled

- similar entities are collected in an **entity set**, e.g., the set of all books, the set of all cars

- an entity is described by a set of pertaining properties (attributes), e.g., each book has an ISBN number, an author, a publisher, ...

- The values of an attribute are from domains like *integer, real, string, ...* e.g., the name of an author is of type *string*

- formal: attribute as a function of an entity onto a codomain e.g., name : author → *string*

- a minimal set of attributes whose values uniquely characterize the associated entity among all entities of its type is called **key**, e.g., ISBN number identifies a book, an article number an article
2. Conceptual Database Design

2.2 The Entity-Relationship Data Model

**Relationship**

- a **relationship** describes a connection between several entities, e.g., student Smith **attends** lecture COP 4720, teaching assistant Benson **works for** professor Meyer

- a homogeneous set of relationships is collected in a **relationship set**, e.g., relationship sets **attends_lecture** or **works_for**

- formal: relationship set \( R \) between the entity sets \( E_1, E_2, \ldots, E_n \) as a relation, i.e.,

  \[
  R \subseteq E_1 \times E_2 \times \cdots \times E_n,\ n\ \text{degree of relationship set }\ R
  \]

  \( \text{attends_lecture} \subseteq \text{students} \times \text{lectures} \)

  \( \text{works_for} \subseteq \text{TAs} \times \text{professors} \)

- attributes may characterize relationships, e.g.
  frequency as an attribute for **attends_lecture**

- an entity set can occur more than once in a relationship set

- if there is only one entity set \( E \) participating in a binary relationship \( R(E, E) \), each of these entity sets can be assigned **roles**
  e.g., \( \text{is_precondition_of} \subseteq \text{lectures} \times \text{lectures} \)
  first lecture / second lecture has the role of a predecessor / successor