Exercise 1.2

- What is a Database (DB)?
  - Collection of related data
  - Represents some aspects of the real world
  - Data is logically coherent
  - Is provided for...
    - ...an intended group of users and
    - ...applications

Exercise 1.3

1. Are the following features provided by a FS, a DB or both?
   - Simple and fast data access: FS, DB
   - Controlled redundancy: DB
   - Sophisticated enforcement of standards: DB
   - Backup and recovery: DB

Exercise 1.4

1. What is redundancy?
   - The same information is stored several times
   - Stored information could be derived from already existing data
2. What are the two main disadvantages of redundancy?
   - Waste of storage space
   - Problems in consistently updating data
3. What is the advantage of redundancy?
   - Speed up data access
   - Use case: For each railroad station and train, precompute the list of all reasonable connecting trains! (This information will be accessed very often but changes rarely)

Exercise 1.5

Which users groups interact with a running DBMS?
   - Database administrators
   - Database designers
   - Application programmers
   - DBMS designers and implementers
   - Tool developers
   - Operators and maintenance personnel
   - End users (naive, sophisticated, casual)
Exercise 1.6

Briefly define the following terms:

– Declarative Querying
  • Specify what you want, not how and from where to get it
  • No need to know position of files etc.
  • No need to know implementation details

– View
  • Create a different perspective of the data by
    – Restricting the accessible information
    – Deriving additional, virtual information
  • This allows for customization of the accessible data for
    specific user-groups

Data Modeling

• Introduction
• Basic ER Modeling
  – Chen notation
  – Alternative notations
• Example

I. Database Applications

• Planning and developing application programs traditionally is a software engineering problem
  – Requirements Engineering
  – Conceptional Design
  – Application Design
  – ...
• Software engineers and data engineers cooperate tightly in planning the need, use and flow of data
  – Data modeling
  – Database design

I. Universe of Discourse

• DB Design models a miniworld into a formal representations
  – Restricted view on the real world with respect to the problems that the current application should solve
1. Phases of DB Design

- **Conceptual Design**
  - Transforms data requirements to conceptual model
  - Conceptual model describes data entities, relationships, constraints, etc. on high-level
    - Does not contain any implementation details
    - Independent of used software and hardware

- **Logical Design**
  - Maps the conceptual data model to the logical data model used by the DBMS
    - e.g. relational model, hierarchical model, ...
    - Technology independent conceptual model is adapted to the used DBMS software

- **Physical Design**
  - Creates internal structures needed to efficiently store/manage data
    - Table spaces, indexes, access paths, ...
    - Depends on used hardware and DBMS software

2. ER Modeling

- **Traditional approach to Conceptual Modeling**
  - Entity-Relationship Models (ER-Models)
    - Also known as Entity-Relationship Diagrams (ERD)
    - Introduced in 1976 by Peter Chen
    - Graphical representation

- **Top-Down-Approach for modeling**
  - Entities and Attributes
  - Relationships
  - Constraints

- **Some derivatives became popular**
  - ER Crow’s Foot Notation (Bachman Notation)
  - ER Baker Notation
  - Later: Unified Modeling Language (UML)
2. ER – Attributes

• Attributes
  – A property of an entity, entity type or a relationship type.
  – Example: name of an employee, color of a car, balance of an account, location of a house, ...
  – Attributes can be classified as being:
    • simple or composite
    • single-valued or multi-valued
    • stored or derived
    • Example: name of a cat is simple, single-valued, and stored

2. ER – Entity Types

• Entity Types are sets of entities sharing the same characteristics or attributes
  – Each entity within the set has its own values
  – Each entity type is described by its name and attributes
  – Each entity is an instance of an entity type
  – Describes the so called schema or intension of similar entities

2. ER – Entity Sets

• An Entity Set is the collection of all entities of a given entity type
  – Entity sets often have the same name as the entity type
    • Cat may refer to the entity type as well as to the set of all Cat entities (sometimes also plural for the set: Cats)
  – Also called the extension of an entity type (or instance)

2. ER – Entity Sets

• Textual Representation
  – Entity Types
    • Written as: EntityName (attribute1, attribute2, …)
  – Entity
    • Written as: (value of attribute1, value of attribute2, …)

• Example
  – Entity Type Cat
    • Cat (name, color)
  – Entity Set Cats
    • (Fluffy, black-white)
    • (Snowflake, white)
    • (Captain Hook, red)
    • …

2. ER – Composite Attributes

• Simple Attribute:
  – Attribute composed of a single component with an independent existence
  – Example: name of a cat, salary of an employee, ...
  – Cat (name, Employee(salary),...)

• Composite Attribute:
  – Attribute composed of multiple components, each with an independent existence
  – Graphically represented by connecting sub-attributes to main attribute
  – Textually represented by grouping sub-attributes in ()
  – Example: address attribute of a company (is composed of street, house number, ZIP, city, and country)
    • Company (address(street, houseNo, ZIP, city))
2. ER Multi-Valued Attributes

- **Single-Valued Attribute**
  - Attribute holding a single value for each occurrence of an entity type
  - Example: name of a cat, registrationNo. of a student

- **Multi-Valued Attributes (lists)**
  - Attribute holding (possibly) multiple values for each occurrence of an entity type.
  - Graphically indicated by a double-bordered oval
  - Textually represented by enclosing in {)
  - Example: telephoneNo. of a student
    - Student (phoneNo)

2. ER - Keys

- Entities are only described by attribute values
  - Two entities with identical values cannot be distinguished (no OIDs, row IDs, etc.)

- Entities (usually) must be distinguishable

- Identification of entities with key attributes
  - Value combination of key attributes is unique within all possible extensions of the entity types
  - Key attributes are indicated by underlining the attribute name

2. ER Modeling

- Example Entity Type
  - Book isbn, author(firstName, lastName)), title, subtitle, publisher(name, city, country), {revision(no, year)}
  - (0321204484, ([Ramez, Elmasri], [Shamkant, Navathe]), Fundamentals of Database Systems, [Pearson, Boston, US], {(2, 1994)})

2. ER - Domains

- Attributes cannot have arbitrary values: they are restricted by the attribute value sets (domains)
  - Zip Codes may be restricted to integer values between 0 and 99999
  - Names may be restricted to character strings with maximum length of 120
  - Domains are not displayed in ER diagrams
  - Usually, popular data types are used to describe domains in data modeling
    - e.g. integer, float, string, ...
2. ER - Domains

• Commonly used data types

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>integer</td>
<td>32-bit signed integer values between $-2^{31}$ and $2^{31}$</td>
</tr>
<tr>
<td>double</td>
<td>double</td>
<td>64-bit floating point values of approximate precision</td>
</tr>
<tr>
<td>numeric</td>
<td>numeric(x, s)</td>
<td>A number with x digit before the decimal and s digits after the decimal</td>
</tr>
<tr>
<td>character</td>
<td>char(x)</td>
<td>A textual string of the exact length x</td>
</tr>
<tr>
<td>varying character</td>
<td>varchar(x)</td>
<td>A textual string of the maximum length x</td>
</tr>
<tr>
<td>date</td>
<td>date</td>
<td>Stores year, month, and day</td>
</tr>
<tr>
<td>time</td>
<td>time</td>
<td>Stores hour, minute, and second values</td>
</tr>
</tbody>
</table>

2. ER - Domains

• Using data types for modeling domains is actually a crutch
  – The original intention of domains was modeling all valid values for an attribute
    • Colors: {Red, Blue, Green, Yellow}
  – Using data types is very coarse and more a convenient solution
    • Colors: varchar(30) ???
  – To compensate for the lacking precision, often restrictions are used
    • Colors: varchar(30) restricted to {Red, Blue, Green, Yellow}

2. ER – NULL Values

• Sometimes, an attribute value is not known or an attribute does not apply for an entity
  – This is denoted by the special value NULL
    • So called NULL-value
  – Example: Attribute "universityDegree" of entity Heinz Müller may be NULL if he does not have a degree
  – NULL is usually always allowed for any domain or data type unless explicitly excluded

ER – Relationships

• Entities are not enough to model a miniworld
  – The power to model dependencies and relationships is needed
• In ER, there can be relationships between entities
  – Each relationship instance has a degree
    • i.e. the number of entities is related to
  – A relationship instance may have attributes

2. ER – NULL Values

• What does it mean when you encounter a NULL-value?
  – Attribute is not applicable
    • e.g. attribute "maiden name" when you don’t have one
  – Value is not known
  – Value will be filled in later
  – Value is not important for the current entity
  – Value was just forgotten to set
• Actually there are more than 30 possible interpretations...

ER – Relationships

• Similar to entities, ERDs do not model individual relationships, but relationship types

• Relationship type
  – Named set of all similar relationships with the same attributes and relating to the same entity types

• Relationship set
  – Set of all relationship instances of a certain relationship type
**ER – Relationships**

- Relationships relate entities within the entity sets involved in the relationship type to each other.

---

**ER – Relationship Cardinality**

- Additionally, restrictions on the combinations of entities participating in an entity set are needed.
  - Example: Relationship type “married to”

---

**ER – Relationship Cardinality**

- Cardinalities express how often a specific entity may appear within a relationship set.
  - A specific entity of type A may appear up to once in the relationship set, an entity of type B appears at least once and at most twice.

---

**ER – Relationships**

- Example:
  - There is an ‘ownership’ relation between cats and persons.
  - But more modeling detail is needed:
    - Does every person own a cat? Does every cat have an owner?
    - Can a cat have multiple owners or a person own multiple cats?
    - Since when does a person own some cat?
    - Who owns whom?

---

**ER – Relationships**

- To each entity of type B, one or two entities of type A are related.

---

- If only one symbol / number is used, this is interpreted as (0, 1).
- Minimum and maximum constrains are used for this.
- Cardinility annotations are used for this.
- Restrictions on the combinations of entities participating in an entity set are needed.
- Common Cardinality Expressions:
  - (0, 1): Zero to one instances are bound by the relationship (i.e. relationship end is optional).
  - (1, 1): Exactly one instance is bound.
  - (0,*): Any number of instances may be bound.
  - (1,*): At least one instance is bound (at least one instance up to any number).
  - No annotation is usually interpreted as (0,*).
  - If only one symbol / number s is used, this is interpreted as (0, s).
  - Sometimes, N or M are used instead of *.

---

**ER – Relationship Cardinality**

- No annotation is usually interpreted as (0, *).
- Minimum and maximum constrains possible.
- Cardinality annotations.
- One cardinality annotation per entity type / relationship end.
- Relationship types referring to just one entity type are called recursive.
- Each person entity may only appear once in the “married to” relationship set.
- Unless living in Utah, a restriction should be modeled that each person can only be married to single person at a time.
- Cardinalities express how often a specific entity may appear within a relationship set.
- ‘T o each entity of type B, one or two entities of type A are related”
Example:

- “Each person can only be married to one other person”
- Each entity can only appear in one instance of the “married to” entity set

Note: following this model a person actually can marry him/herself

ER – Relationship Cardinality

- Cardinalities for binary relationship types can be classified into common, more general cardinality types

  - These cardinality types are also often found in other modeling paradigms
    - 1:1 (One-To-One) – Each entity of the first type can only relate to exactly one entity of the other type
    - 1:N (One-To-Many) – Each entity of the first type can relate to multiple entities of the other type
    - N:1 (Many-To-One) – Multiple entities of the first type can relate to exactly one entity of the second type
    - N:M (Many-To-Many) – No restrictions. Any number of entities of first type may relate to any number of entities of second type.

ER – Relationship Roles

- Often, it is beneficial to clarify the role of an entity within a relationship

  - Example: Relationship “supervises”
    - What is meant? Who is the supervisor? Who is the supervised person?
    - Roles can be annotated on the relationship lines
**ER – Relationship Degree**

- Relationship instances involve multiple entities
  - The number of entities in each relationship instance is called relationship degree
    - \( \text{Degree}=2 \): Binary Relation
    - \( \text{Degree}=3 \): Ternary Relation

**ER – Relationship Attributes**

- Similar to entities, relationship types may even have attributes
  - For 1:1 relationships, the relationship attribute may be migrated to any of the participating attributes
  - For 1:N relationships, the attribute may be only migrated to the entity type on the N-side
  - For N:M relationships, no migration is possible

**ER – Total Participation**

- To express that all entities of an entity type appear in a certain relationship set, the concept of total participation can be used
  - The entity type which is totally participating is indicated by a double line
  - Example: “Each driver’s license must belong to a person”

**ER – Weak Entities**

- Each entity needs to be identifiable by a set of key attributes
  - Entities existing independently of the context are called strong entities
    - A person exists whether it is married or not
  - In contrast, there may be entities without an unique key called weak entities

- Weak entities are identified by being related to strong entities
  - The strong entities “own” the weak entity
    - The weak one cannot exist without the strong ones
  - The relationships relating the strong to the weak are called identifying relationships
    - The weak entity is always totally participating in that relationship
  - Weak entities have partial keys which are unique within the identifying relationship sets of their strong entities
    - To be unique, the weak entity instance has to borrow the keys of the respective strong entity instances

- Weak entity types and identifying relationship types are depicted by double-lined rectangles
  - Example:
    - “An online shopping order contains several order items”
    - An order item can only exits within an order
    - Each order item can be identified by the orderNo of it’s owning order and its itemLine
ER – Overview

- Entity Type
- Weak Entity Type
- Attribute
- Key Attribute
- Multi-valued Attribute
- Composite Attribute
- Derived Attribute
- Relationship Type
- Identifying Rel. Type

ER – Overview

- Total participation of E2 in R
- Cardinality
  - An instance of E1 may relate to multiple instances of E2
- Specific cardinality with min and max
  - An instance of E1 may relate to multiple instances of E2

ER – Alternative Notations

- There is a plethora of alternative notations for ER diagrams
  - Different styles for entities, relationships and attributes
  - No standardization among them
  - Also, notations are often freely mixed
    - ER diagrams can look completely different depending on the used tool / book
  - In the following, we introduce the (somewhat popular) crow’s feet notation

ER – Crow’s Foot Notation

- Entity Types
  - Entity Types are modeled with a named box
  - Attribute names are written inside the box separated by a line
    - Key attributes are marked with a leading asterisk
    - Composite attributes are represented with indentation

ER – Crow’s Foot Notation

- Relationship Types
  - Relationship types are modeled by lines connecting the entities
  - Line is annotated with the name of the relationship which is a verb
  - Cardinalities are represented graphically
    - (0, 1) : Zero or one
    - (1, 1) : Exactly one
    - (0, *) : Zero or more
    - (1, *) : one or more
    - ATTENTION: Cardinalities are written on the opposite side of the relationship (in contrast to “classic ER”)
What happens to n-ary relationships or relationship attributes?

Problem

- N-ary relationship types are not supported by crow’s foot notation, neither are relationship attributes

Workaround solution:

- Intermediate entities must be used

N-ary relationships are broken down in a series of binary relationship types anchoring on the intermediate entity

Originally, ER diagrams were intended to be used on a conceptual level

- Model data in an abstract fashion independent of implementation

Crow’s foot notation sacrifices some conceptual expressiveness

- Model is closer to the logical model (i.e. the way the data is later really stored in a system)
- This is not always desirable and may obfuscate the intended semantics of the model

Barker’s notation

- Based on Crow’s Feet Notation
- Developed by Richard Barker for Oracle’s CASE modeling books and tools in 1986
- Cardinalities are represented differently
  - (0, 1): Zero or one
  - (1, N): Zero or more
  - (1, N): one or more
- Cardinalities position similar to Crow’s Foot notation and opposite to classic ER
- Different notation of subtypes
  - See next lecture

Black Diamond Notation

- Cardinalities are represented differently
  - Cardinality annotation per relationship, not per relationship end
  - 1:1
  - 1:N
  - N:M
- Also, N-ary relationships possible
  - ternary
### 2. ER – Mathematical Model

- Mathematically, an **attribute** $A$ of **entity type** $E$ with **domain** $V$ is a function from $E$ to the power set $P(V)$
  - $A : E \rightarrow P(V)$
    - The power set $P(V)$ of $V$ is the set of all subsets of $V$
    - The **value** of an attribute of the entity $e$ is denoted as $A(e)$
    - This definition covers
      - null values (empty set)
      - single-valued attributes (restricted to singleton sets)
      - multi-valued attributes (no restrictions)
    - For a composite attribute $A(A_1, A_2, \ldots, A_n), V$ is defined as
      - $V = P(V_1) \times P(V_2) \times \ldots \times P(V_n)$

### An example

- **We want to model a simple university database**
  - In our database, we have students. They have a name, a registration number, and a course of study.
  - The university offers lectures. Each lecture may be part of some course of study in a certain semester. Lectures may have other lectures as prerequisites. They have a title, provide a specific number of credits and have an unique ID
  - Each year, some of these lectures are offered by a professor at a certain day at a fixed time in a specific room. Students may register for that lecture.
  - Professors have a name and are member of a specific department.

### 2. ER – Mathematical Model

- A **relationship type** $R$ among $n$ entity types $E_1, E_2, \ldots, E_n$ defines a **relationship set** among instances of these entity types
  - Each relationship instance $r_i$ within the relationship set $R$ associates $n$ individual entities $(e_{i1}, e_{i2}, \ldots, e_{in})$, and each entity $e_{ij}$ in $r_i$ is member of the entity type $E_j, 1 \leq j \leq n$
  - Alternatively, the relationship type $R$ can be seen as a subset of the Cartesian product of the entity types
    - $R \subseteq E_1 \times E_2 \times \ldots \times E_n$

### An example

- **How to start? What to do?**
  - Find the basic **entity types**
  - Find the **attributes** of entities
    - Decide to which entity an attribute should be assigned
    - Which attributes are key attributes?
    - Some attributes are better modeled as own entities, which ones?
  - Define the **relationship types**
    - Which role do entities play?
    - Do relationships require additional entity types?
    - Are the relationships total? Identifying? Are weak entities involved?
    - What are the cardinalities of the relationship type?

### An example

- **Which are our entity types?**
  - In our database, we have **students**. They have a name, a registration number and a course of study.
  - The university offers **lectures**. Each lecture may be part of some course of study in a certain semester. Lectures may have other lectures as prerequisites. They have a title, provide a specific number of credits and have unique ID
  - Each year, some of these lectures are offered by a **professor** at a certain day at a fixed time in a specific room. Students may attend that lecture.
  - Professors have a name and are member of a specific department.

### An example

- **What attributes are there?**
  - In our database, we have students. They have a name, a registration number and a course of study.
  - The university offers lectures. Each lecture may be part of some course of study in a certain semester. Lectures may have other lectures as prerequisites. They have a **credit**, provide a specific number of credits and have unique ID
  - Professors have a name and are member of a specific department.

---

*2. ER – Mathematical Model*

- Mathematically, an attribute $A$ of entity type $E$ with domain $V$ is a function from $E$ to the power set $P(V)$
  - $A : E \rightarrow P(V)$
    - The power set $P(V)$ of $V$ is the set of all subsets of $V$
    - The value of an attribute of the entity $e$ is denoted as $A(e)$
    - This definition covers
      - null values (empty set)
      - single-valued attributes (restricted to singleton sets)
      - multi-valued attributes (no restrictions)
    - For a composite attribute $A(A_1, A_2, \ldots, A_n), V$ is defined as
      - $V = P(V_1) \times P(V_2) \times \ldots \times P(V_n)$

*An example*

- We want to model a simple university database
  - In our database, we have students. They have a name, a registration number, and a course of study.
  - The university offers lectures. Each lecture may be part of some course of study in a certain semester. Lectures may have other lectures as prerequisites. They have a title, provide a specific number of credits and have an unique ID
  - Each year, some of these lectures are offered by a professor at a certain day at a fixed time in a specific room. Students may register for that lecture.
  - Professors have a name and are member of a specific department.

*2. ER – Mathematical Model*

- A relationship type $R$ among $n$ entity types $E_1, E_2, \ldots, E_n$ defines a relationship set among instances of these entity types
  - Each relationship instance $r_i$ within the relationship set $R$ associates $n$ individual entities $(e_{i1}, e_{i2}, \ldots, e_{in})$, and each entity $e_{ij}$ in $r_i$ is member of the entity type $E_j, 1 \leq j \leq n$
  - Alternatively, the relationship type $R$ can be seen as a subset of the Cartesian product of the entity types
    - $R \subseteq E_1 \times E_2 \times \ldots \times E_n$

*An example*

- How to start? What to do?
  - Find the basic entity types
  - Find the attributes of entities
    - Decide to which entity an attribute should be assigned
    - Which attributes are key attributes?
    - Some attributes are better modeled as own entities, which ones?
  - Define the relationship types
    - Which role do entities play?
    - Do relationships require additional entity types?
    - Are the relationships total? Identifying? Are weak entities involved?
    - What are the cardinalities of the relationship type?

*An example*

- Which are our entity types?
  - In our database, we have students. They have a name, a registration number and a course of study.
  - The university offers lectures. Each lecture may be part of some course of study in a certain semester. Lectures may have other lectures as prerequisites. They have a title, provide a specific number of credits and have unique ID
  - Each year, some of these lectures are offered by a professor at a certain day at a fixed time in a specific room. Students may attend that lecture.
  - Professors have a name and are member of a specific department.

*An example*

- What attributes are there?
  - In our database, we have students. They have a name, a registration number and a course of study.
  - The university offers lectures. Each lecture may be part of some course of study in a certain semester. Lectures may have other lectures as prerequisites. They have a credit, provide a specific number of credits and have unique ID
  - Professors have a name and are member of a specific department.
**First try…**
- This model is **really crappy**
- "Course of study" does not seem to be an attribute
- "Professor" does not have key attributes

**Second try**
- Professors use a new entity type
- "Prerequisite lecture" also is not a good attribute
- "Course of study" does not seem to be an attribute

**Not really intuitive…**
- Use an intermediate entity instead?

**Better?**
- Add cardinalities
- Add total and identifying annotations

**The same example using Crow’s Foot Notation**
- Note that the “part of” relationship had to use an intermediate entity
• Modeling is not that simple
• Many possible (and also correct) ways of modeling the same miniworld
  – Some are more elegant, some are less elegant
• Models alone are not enough, they need to be documented
  – What are the meanings of the attributes? The meanings of the relationships?