Chapter 2: ER-Diagrams

Content:

• Learn how to draw ER diagrams
Database Design

DBS can take care automatically of many things – but the user has to specify

- Requirements of the application
- Characteristics of the data

Two important concepts during DBS design:
- Data Model: How to describe the data?
- Data Schema: Concrete description of the data (using the chosen data model)
Data modeling

Excerpt of the Real World

Manual/intellectual Modeling

Conceptual Schema (E/R- or UML-Schema)

Semi-automatic Transformation

Relational Schema

XML Schema

Network Schema

Object-orientierten Schema
Modeling a small example application: E/R

Real World: University

Conceptual Modeling

Students
- StudNr
- Name

Lectures
- LectureNr
- Title

attend

requires
### Relational Data Model

<table>
<thead>
<tr>
<th>Students</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>StudNr</td>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>26120</td>
<td>Fichte</td>
<td></td>
</tr>
<tr>
<td>25403</td>
<td>Jonas</td>
<td></td>
</tr>
<tr>
<td>25403</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26120</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>attend</th>
</tr>
</thead>
<tbody>
<tr>
<td>StudNr</td>
</tr>
<tr>
<td>25403</td>
</tr>
<tr>
<td>26120</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lectures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lecture Nr</td>
</tr>
<tr>
<td>5001</td>
</tr>
<tr>
<td>5022</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

**Select** Name  
**From** Students, attend, Lectures  
**Where** Students.StudNr = attend.StudNr and attend.LectureNr = Lectures.LectureNr and Lectures.Title = 'Grundzüge';
Logical Data Models

- Network Model
- Hierarchical Model
- **Relational Data Model**
- XML Model
- Object-oriented Data Model
  - Object-relational Schema
- Deductive Data Model

* [Michael Stonebraker: What Goes Around Comes Around]
Modeling a small example application: UML

<table>
<thead>
<tr>
<th>Students</th>
<th>+Attendee 1..*</th>
<th>+Successor *</th>
</tr>
</thead>
<tbody>
<tr>
<td>+StudNr : int</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+Name : String</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+Semester : int</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+GPA() : float</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+SumWeeklyHours() : short</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lectures</th>
<th>requires *</th>
</tr>
</thead>
<tbody>
<tr>
<td>+LectureNr : int</td>
<td></td>
</tr>
<tr>
<td>+Title : String</td>
<td></td>
</tr>
<tr>
<td>+WeeklyHours : int</td>
<td></td>
</tr>
<tr>
<td>+NumberAttendees() : int</td>
<td></td>
</tr>
<tr>
<td>+FailureRate() : float</td>
<td></td>
</tr>
</tbody>
</table>
Phases of Database Design

- Information Requirements
- Requirements Engineering
- Conceptual Modeling
  - ER Schema
  - Conceptual Design
- Logical Modeling
  - Logical Schema
  - Logical Design
- Physical Modeling
  - Physical Database Design

- Processing Requirements
- DBMS-Characteristics
- Hardware/OS-Characteristics
Software Development and Ability to Communicate

- How the customer explained it
- How the Project Leader understood it
- How the Analyst designed it
- How the Programmer wrote it
- How the Business Consultant described it
- How the project was documented
- What operations installed
- How the customer was billed
- How it was supported
- What the customer really needed
Create a "Scope Statement" consisting of:

- Entity description
- Relation description
- Process description
University Employees

-Quantity: 1000
-Attributes
  - **EmpNumber**
    - Type: Integer
    - Domain: 0...999.999.99
    - Defined: 100%
    - Identifying: yes
    - Example: 007
  - **Salary**
    - Type: decimal
    - Length: (7,2)
    - Unit: Euro per month
    - Defined: 10%
    - Identifying: no
  - **Level**
    - Type: String
    - Length: 2
    - Defined: 100%
    - Identifying: no
    - Example: W2
Relation Description: exam

Involved Objects:
- Professor as Tester
- Student as Testee
- Lecture as Test Subject

Attributes of the Relation:
- Date
- Time
- Grade

Quantity: 100 000 per year
Process Description: 

**Issue a Certificate**

- Frequency: semiannually
- Required Data
  * Tests
  * Examination Rules
  * Student’s Records
  * ...
- Priority: high
- Data Volume to be processed
  * 500 Students
  * 3000 Tests
  * 10 Versions of Examination Rules
Creating a Specification

The actual analysis is an iterative process:
• Customer tells developer his/her needs
• Developer notes everything down (s/he understood) in his/her „language” . . .
• . . . and translates it into the "language" of the customer
• This is shown to the customer who does not agree with everything
• Change requests are agreed on
• Back to step 2
Phases of Database Design

1. Information Requirements
2. Requirements Engineering
3. Conceptual Modeling
4. Logical Modeling
5. Physical Modeling
6. ER Schema
7. Conceptual Design
8. Logical Design
9. Physical Database Design
10. Processing Requirements
11. DBMS-Characteristics
12. Hardware/OS-Characteristics

Relation

<table>
<thead>
<tr>
<th>Att1</th>
<th>Att2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The ideal design (the ideal specification) is
• unique
• complete
• comprehensible (for all participants)
• nonredundant
• . . . and not reachable in reality
Entity/Relationship-Modeling

Entity
- Students
  - StudNr
  - Name
  - Semester
  - Attendee

Relationship
- attend

Attribute (property)
- LectureNr
- Title
- Weekly hours

Key (identification)
- Name

Role
- Attendee
Entity/Relationship-Modeling

Mathematically: “Relational Schema”

- **Entities** are sets of n-ary tuples:
  - Students = \{[1, "Sam", 3],
                [2, "Jack", 5], ...
  \}

- **Relationships** are n-ary relations:
  - attend \subseteq Students \times Lectures
    = \{[1, 101], [1, 102],
         [2, 101]\}
Functionalities

$R \subseteq E_1 \times E_2$

1:1

1:N

N:1

N:M
Relationship 1:1

One \( e_1 \in E_1 \) has 0 or 1 partners in \( E_2 \)
One \( e_2 \in E_2 \) has 0 or 1 partners in \( E_1 \)

Example:

one car has one license plate
one license plate belongs to one car
Relationship 1:N

One \( e_1 \in E_1 \) has \( N \) partners in \( E_2 \)
One \( e_2 \in E_2 \) has 0 or 1 partners in \( E_1 \)

Example:

one mentor advises several students
one student is advised by one mentor
One \( e_1 \in E_1 \) has \( N \) partners in \( E_2 \)
One \( e_2 \in E_2 \) has \( N \) partners in \( E_1 \)

Example:

one actor stars in several movies
one movie has several actors
Recursive Relationship 1:N

Relationship 1:N

One $e_1 \in E_1$ is called ‘roleA’ and has N partners in $E_1$
One $e_2 \in E_1$ is called ‘roleB’ and has 0 or 1 partners in $E_1$

Example:

One person is the mother of several persons (children)
One person is the child of one person (mother)
Functionalities in \( n \)-ary Relationships

\[
R : E_1 \times E_2 \times \ldots \times E_n \rightarrow E_k
\]
Example Seminar

supervise : Topics x Students → Professors

supervise : Professors x Students → Topics
Student x Professor -> Topic

1. A Student is only allowed to work on one topic with a given professor.
2. Students may work on the same topic only once – thus they may not work on the same topic again with another professor.
Not: Professor x Topic -> Student

3. Professors can reuse the same topic – i.e., one professor can give the same topic to different students. (absence of: PXT -> S)
Thereby induced Consistency Constraints

1. A Student is only allowed to work on one topic with a given professor. (SxP -> T)

2. Students may work on the same topic only once – thus they may not work on the same topic again with another professor. (SxT -> P)

3. Professors can reuse the same topic – i.e., one professor can give the same topic to different students. (absence of: PxT -> S)

4. The same topic can be given by different professors – but to different students. (absence of: PxT -> S)

5. The same professor can give different topics – but to different students. (absence of: PxT -> S)
Occurrence of the Relationship supervise

Dashed lines represent illegal occurrences
University Schema

- **Students**
  - StudNr
  - Name
  - Semester

- **Lectures**
  - LectureNr
  - Title
  - Weekly hours

- **Assistants**
  - PersNr
  - Name
  - Area

- **Professors**
  - PersNr
  - Name
  - Level
  - Room

Relationships:
- Students attend Lectures
- Lectures require follow-up
- Prerequisite:
  - Students prerequisite Lectures

- Assistants work-for Professors
- Grade given by Professors on Lectures

- Grade prerequisite for Professors
- Name prerequisite for Assistants
(min, max)-Notation

For every $e_i \in E_i$ there are
- at least $\min_i$ tuples $(..., e_i, ...) \in R$ and
- at most $\max_i$ tuples $(..., e_i, ...) \in R$
Example (min, max)

One mentor advises up to 20 students.
One student is advised by exactly one mentor.
Min, max Notation and Functionalities

A Polygon has at least 3 Edges. An Edge has 1 or 2 Polygons.
Weak Entities

- Relationship between "strong" and "weak" type is 1:N (or 1:1 in rare cases) - why not N:M?
- The existence of a room depends on the existence of the associated building
- RoomNr is unique only within the building
- Key of Rooms is: RoomNr and BldNr
Tests as weak entity type

- Several professors design one test
- Several lectures are inquired in one test
Generalization / Specialization:

S is a specialization of R

Example:

animal

Is-a

dog

name

pet_id

breed

color
Generalization University

University-Members

-is-a-

Students

StudNr

Name

Employees

PersNr

-is-a-

Assistents

Professors

Area

Level

Room
Conclusion

University schema with generalization and (min, max)-notation
Where are we?

Information Requirements

Requirements Engineering

Scope Statement

Conceptual Modeling

ER Schema

Conceptual Design

Logical Modeling

Logical Schema

Logical Design

Physical Modeling

Physical Database Design

Processing Requirements

DBMS-Characteristics

Hardware/OS-Characteristics

R := [{att1, att2}]

<table>
<thead>
<tr>
<th>Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Att1</td>
</tr>
<tr>
<td>Att2</td>
</tr>
</tbody>
</table>