DATABASE DESIGN I - 1DL300

Spring 2013

An Introductory Course on
Database Systems

http://www.it.uu.se/edu/course/homepage/dbastekn/vt13/

Uppsala Database Laboratory
Department of Information Technology, Uppsala University,
Uppsala, Sweden
Introduction to Database Design Using Entity-Relationship Modeling

Elmasri/Navathe ch 7, 9
Padron-McCarthy/Risch ch 2-3

Sobhan Badiozamany
Silvia Stefanova

Department of Information Technology
Uppsala University, Uppsala, Sweden
Outline

1. An example
2. Database design process
3. E-R Model
4. To Summarize
Design the database *University*
DB *University*, Requirements:

1. Store the departments’ names, the subdivisions names, and the departments’ addresses.
2. Store the names and the personal numbers of the teachers. Each teacher is employed in one department.
3. Each subdivision has own courses.
4. A teacher can teach several courses.
DB Teachers

• Are there any problems with the proposed design?
• Is it a “good” design?

• Think on
  • Clear concepts (clear semantics)?
  • Needed memory for storing?
  • Update (insert, update, delete)?
  • Search?
### DB Teachers

#### Courses

<table>
<thead>
<tr>
<th>Subdivision</th>
<th>pnumber</th>
<th>teacher</th>
<th>course</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Systems</td>
<td>1234</td>
<td>Sara</td>
<td>Database I</td>
</tr>
<tr>
<td>Electrical Engineering</td>
<td>4567</td>
<td>Erik</td>
<td>Electrical circuits</td>
</tr>
<tr>
<td>Systems and Control</td>
<td>1234</td>
<td>Sara</td>
<td>Control systems</td>
</tr>
</tbody>
</table>

#### Divisions

<table>
<thead>
<tr>
<th>subdivision</th>
<th>department</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Systems</td>
<td>Informational Technology</td>
<td>Polack</td>
</tr>
<tr>
<td>Systems and Control</td>
<td>Informational Technology</td>
<td>Polack</td>
</tr>
<tr>
<td>Electrical Engineering</td>
<td>Engineering</td>
<td>Ångström</td>
</tr>
</tbody>
</table>
To remember!

What we *don’t want*:  
- Anomalies on update, insertion, deletion  
- Redundancy (do things several times)  
- Errors in our data (violation of integrity)  
- Loss of data  

What we *do want*:  
- Easy work with data: Do things one time (i.e. update, deletion)!  
- Should be logical, easy to understand.  
- Assurance: Data is correct, doesn’t get lost  

**Conclusion:** Systematic Design Process is needed
Outline

1. An example
2. Database design process
3. E-R Model
4. To Summarize
Database Design Process

- Requirements Collection & Analysis
- Conceptual Design
  - Conceptual schema
  - Logical schema
- Logical Design
- Physical Design
  - Internal schema

Conceptual model
- E-R model

Implementation model
- Model of DBMS
  - (Relational model)
ER-modeling

• High-level conceptual specification of the content in the database.

• History

• Why ER-models?
  o High-level description - easier to understand for non-technicians
  o More formal than natural language - avoid misconceptions and multiple interpretations
  o Implementation independent (of DBMS) - less technical details
  o Documentation
  o Model transformation to an implementation data model
Outline

1. An example
2. Database design process
3. E-R Model
4. To Summarize
Entity type and entity

- **Entity**: Thing in the real world, e.g. *Erik Svensson*, ...
- **Attributes**: properties describing the entity
- **Entity Type**: collection (set) of entities that have the same attributes, e.g. *Person, Car, Hospital*
- **Entity types** express the intention, i.e. the meaning of the concept whereas the **set of entities** represents the extension of that type.
Attributes

• Every attribute has a **domain** (or value set).
  - A domain specifies the set of allowed values each individual attribute can be assigned.

• There is (at least) six different types of attributes:
  - simple
    - sex: M or F
  - composite
    - name: (Igor, Karlsson)
  - single-valued
    - name: “Igor Karlsson”
  - multivalued
    - friends: {Nasse, Puh,...}
  - stored
    - birthdate: 980917
  - derived
    - age: 0
  - null
Identification of Entities

- **Key attribute(s):** attribute(s) that has unique values for every instance of an entity type
- An entity type can have more than one key.
Relationship type and relationship

- **Relationship**: relation (association) between entities
- **Relationship type**: represents a relationship between a number of entity types.
- A relationship type \( R \) is a set of relationships (i.e. relational instances) or tuples.
- A relationship type, \( R \), can mathematically be defined as: \( R \subseteq E_1 \times E_2 \times \ldots \times E_n \), \( E_j \) is a entity type.
- E-R representation: diamond-shaped box
Relationship type and relationship

E1(student): {Jan, Kalle, Anna}   E2(course): {Databases, Algorithms}

E1 × E2: { (Jan, Databases), (Jan, Algorithms),  
           (Kalle, Database)s, (Kalle, Algorithms)  
           (Anna, Databases), (Anna, Algorithms) }  

R ⊆ E1 × E2 : { (Jan, Databases), (Jan, Algorithms),  
               (Anne, Databases) }
Cardinality ratio of relationship types

- **Cardinality ratio constraint** specifies the number of relational instances that an entity can take part in.

For binary relationship types:
- one-to-one (1:1)
- one-to-many (1:N)
- many-to-many (M:N)
Structural constraints for relationship types

- **Participation constraint**
  - Specifies whether the entity existence is dependent of another entity via a relationship type
  - **Partial participation**: the entity can exist without this relationship
  - **Total participation**: the entity requires this relationship in order to exist.
Roles of relational types

• A role name specifies what **role** an entity type plays in a specific relationship
• Especially for **recursive relationships**
Attributes for relationship types

- A relationship type can have attributes
- If the relation is a 1:1 or 1:N relation, the attribute can be stored at one of the participating entities.
- When the relation is of the type M:N one must store the attributes with the instance of the relation.

```
<table>
<thead>
<tr>
<th></th>
<th>M</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Works_on</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

- Working hours
Attributes for relationship types – Transform!

- **Employee**
  - **Works_on**
  - **Project**
    - **Working hours**

- **Master student**
  - **Works_on**
  - **Master thesis**
  - **Supervisor**

- **Student**
  - **Study**
    - **Course**
    - **Percenatge**
Weak entity type

- **Weak entity types**: those that are meaningless without an owner entity type.
- Do not have an own key but **partial key**
- The relationship to the owner is called the *identifying relationship*.
ER-notation (Elmasri/Navathe Figure 7.14)
**Task:** Design an ER diagram for the University database!
• An enterprise consists of a number of departments. Each department has a name, a number, a manager, and a number of employees. The starting date for every department manager should also be registered. A department can have several office rooms.

• Every department finances a number of projects. Each project has a name, a number and an office room.

• For each employee, the following information is kept: name, social security number, address, salary and sex. An employee works for only one department but can work with several projects that can be related to different departments. Information about the number of hours (per week) that an employee work with a project should be stored. Information about the employees manager should also be stored.
Example ER-modeling 2

**ER Diagram:**

- **EMPLOYEE**
  - fstname
  - name
  - sex
  - salary
  - ssn
  - famname
  - address

- **IS-MANAGER**
  - 1:1

- **WORKS-FOR**
  - EMPLOYEE
  - DEPARTMENT
  - 1:N

- **MANAGES**
  - EMPLOYEE
  - FINANCES
  - 1:M

- **WORKS-WITH**
  - EMPLOYEE
  - PROJECT
  - 1:N

- **DEPARTMENT**
  - name
  - number
  - room

- **FINANCES**
  - startdate
  - number

- **PROJECT**
  - name
  - number
  - room
  - hours
Recovering multi-valued attributes
(If we want only single valued attributes)
ER model transformations

Replacing multi-valued attributes \textit{by an entity type}
ER model transformations

Replacing M-N relationships
ER model transformations

Replacing M-N relationships with a weak entity type and binary relationships.
ER model transformations

Replacing M-N relationships with a regular entity type and binary relationships.
Subclasses, superclasses & inheritance

- Two generic ideas for creating superclass/subclass relationships
  - Specialization of superclass into subclasses
  - Generalization of subclasses into a superclass (Elmasri/Navathe Figure 7.21)
Subclasses, superclasses & inheritance

- Disjointness (Elmasri/Navathe Figure 7.23)
Subclasses, superclasses & inheritance

- Overlapping (non-disjoint) of subclasses (Elmasri/Navathe Figure 7.23)
Aggregation: an abstract concept for building composite objects from their component objects

- Aggregate attribute values of an object to form the whole object
- Represent an aggregation relationship as an ordinary relationship

![ER Diagram]

- Represent an aggregation relationship as an ordinary relationship

![Another ER Diagram]
Aggregation in ER

✓ Combining objects related by a particular relationship instance into a *higher-level aggregate object* (Elmasri/Navathe fig. 7.28)

![ER Diagram](image-url)
Aggregation in ER

✓ Combining objects related by a particular relationship instance into a higher-level aggregate object (Elmasri/Navathe fig. 7.28)
Summary

• Database design process
• E-R model
  o Entity and entity type
  o Relationship and relationship type
  o Cardinality ratio and structural constraints for relationship types
  o Attributes for relationship types
  o Weak entity type
  o E-R transformations
  o Inheritance
  o Aggregation in ER