Course A7B36DBS: Database Systems

Lecture 01:
Conceptual Database Modeling

Martin Svoboda
Irena Holubová
Tomáš Skopal

Faculty of Electrical Engineering, Czech Technical University in Prague
Course Plan

• What is this course about?
  ▪ Relational database management systems (RDBMS)
    – Data modeling and database design techniques
    – SQL – data definition and manipulation language
    – Formal query languages for the relational model
    – Basics of physical implementation and transactions
  ▪ It will also slightly introduce you to...
    – Database applications
    – Multimedia, XML, NoSQL and other databases
  ▪ But it is not about...
    – Data mining, data warehouses, OLAP, cloud computing, ...
Course Plan

- Plan of lectures
  - Conceptual database modeling (ER, UML, OCL)
  - Logical database models (relational model)
  - SQL (DDL, DML, ..., SQL/XML)
  - Relational algebra and relational calculus
  - Database design (integrity constraints, normal forms, ...)
  - Database structures (files, index structures, ...)
  - Transactions (scheduling, locking protocols, ...)
  - Database architectures and models
Course Plan

• Plan of practical classes
  ▪ ER and UML conceptual modeling
  ▪ Transformation of ER / UML to the relational model
  ▪ SQL – data definition and updates
  ▪ SQL – querying and views
  ▪ Relational algebra and calculus
  ▪ Functional dependencies and keys
  ▪ Algorithms of decomposition and synthesis
Organizational Stuff

• See the web page of the lecture:

  ▪ Practical classes
    – Structure, attendance, assignments, points, activity, credits

  ▪ Lectures
    – Examination, points, grades
Outline

• **Introduction to database systems**
  - What is a database?
  - Basic terminology

• **Conceptual database modeling**
  - ER – Entity-Relationship Model
  - UML – Unified Modeling Language
  - OCL – Object Constraint Language
Database Systems
Basic Terminology

- **Database (DB)**
  - Logically ordered collection of related data instances
  - Self-describing, meta-data stored together with data
    - Data + schema + integrity constraints

- **Database management system (DBMS)**
  - General software system for access to a database
  - Provides mechanisms to ensure security, reliability, concurrency, integrity of stored data, ...
Brief History

• Database models and systems
  ▪ Network and hierarchical databases
  ▪ Relational databases
  ▪ Object and object-relational databases
  ▪ XML databases
  ▪ NoSQL databases
    – Key-value stores, document-oriented, graph databases, ...
  ▪ Stream, active, deductive, spatial, temporal, probabilistic, real-time, in-memory, embedded, ...

• Still evolving area with plenty of challenges
Brief History

Why so many different database systems?

- **Different contexts**
  - OLTP, OLAP, Cloud computing, Big data, ...

- **Different requirements**
  - Performance, scalability, consistency, availability, ...

- **Different architectures**
  - Centralized, distributed, federated, ...

- **Different forms of data**
  - Relations, objects, graphs, ...
  - Semi-structured, unstructured data, texts, ...
  - Multimedia, web
Motivation for Databases

• Why database systems?
  ▪ Data sharing and reusability
    – Consistency, correctness, ...
    – Elimination of redundancies
    – Concurrency, isolation, transactions, ...
  ▪ Unified interface and languages
    – Data definition and manipulation
  ▪ Information security
    – User authentication, access authorization, ...
  ▪ Administration and maintenance
    – Replication, backup, recovery, migration, tuning, ...
Database Modeling

• Process of database design
  – One vague sentence at the beginning...
  – ... a fully working system at the end
  ▪ Understanding and modeling the reality
  ▪ Organizing the acquired information
  ▪ Balancing the identified requirements
  ▪ Creating a suitable database schema

• Who are stakeholders?
  ▪ Stakeholder is any person which is relevant for your application
    – E.g. application user, investor, owner, domain expert, etc.
Basic Terminology

- **Model** = modeling language
  - Set of constructs you can use to express something
  - UML model = \{class, attribute, association\}
  - Relational model = \{relational schema, attribute\}

- **Schema** = modeling language expression
  - Instance of a model
  - Relational schema = \{Person(name, email)\}

- **Diagram** = schema visualization
Layers of Database Modeling

• **Conceptual layer**
  - Models a part of the reality (problem domain) relevant for a database application, i.e. identifies and describes real-world entities and relationships between them
  - Conceptual models such as ER or UML

• **Logical layer**
  - Specifies how conceptual components are represented in database structures
  - Logical models such as relational, object-relational, graph, ...

• **Physical layer**
  - Specifies how logical database structures are implemented in a specific technical environment
  - Data files, index structures (e.g. B+ trees), etc.
Conceptual Database Modeling
Conceptual Database Modeling

• Conceptual modeling

  ▪ Process of creating a conceptual schema of a given problem domain
    – In a selected modeling language
    – And on the basis of given requirements

  ▪ Multiple conceptual schemas are actually created
    – Each schema describes the given database application(s) from a different point of view
    – Even different conceptual models may be needed

  ▪ We focus only on conceptual data viewpoint
Conceptual Modeling Process

Analyze requirements
• Identify types of entities
• Identify types of relationships
• Identify characteristics

Model identified types
• Choose modeling language
• Create conceptual schema
• Create schema diagram

Iteratively adapt your schema to requirements changing over time
Requirement Analysis (Step 1)

- Step 1 of conceptual modeling
  - Start with requirements of different stakeholders
    - Usually expressed in a natural language
    - Meetings, discussions, inquiries, ...
  - Identify important...
    - types of real-world entities,
    - their characteristics,
    - types of relationships between them, and
    - their characteristics
  - ... and deal with ambiguities
Identification of Entities (Step 1.1)

• Example

  - Try to identify all types of entities:

    Our environment consists of persons which may have other persons as their colleagues. A person can also be a member of several research teams. And, they can work on various research projects. A team consists of persons which mutually cooperate. Each team has a leader who must be an academic professor (assistant, associate or full). A team acts as an individual entity which can cooperate with other teams. Usually, it is formally part of an official institution, e.g., a university department. A project consists of persons working on a project but only as research team members.
Identification of Entities (Step 1.1)

• Example

Our environment consists of persons which may have other persons as their colleagues. A person can also be a member of several research teams. And, they can work on various research projects. A team consists of persons which mutually cooperate. Each team has a leader who must be an academic professor (assistant, associate or full). A team acts as an individual entity which can cooperate with other teams. Usually, it is formally part of an official institution, e.g., a university department. A project consists of persons working on a project but only as research team members.

• Identified entity types

- Person
- Team
- Project
- Professor
  - Assistant Professor
  - Associate Professor
  - Full Professor
- Institution
- Department
Identification of Relationships (Step 1.2)

• Example
  - Try to identify all types of relationships:

    Our environment consists of persons which may have other persons as their colleagues. A person can also be a member of several research teams. And, they can work on various research projects. A team consists of persons which mutually cooperate. Each team has a leader who must be an academic professor (assistant, associate or full). A team acts as an individual entity which can cooperate with other teams. Usually, it is formally part of an official institution, e.g., a university department. A project consists of persons working on a project but only as research team members.
Identification of Relationships (Step 1.2)

• Example

Our environment consists of persons which may have other persons as their colleagues. A person can also be a member of several research teams. And, they (person) can work on various research projects. A team consists of persons which mutually cooperate. Each team has a leader who must be an academic professor (assistant, associate or full). A team acts as an individual entity which can cooperate with other teams. Usually, it (team) is formally part of an official institution, e.g., a university department. A project consists of persons working on a project but only as research team members.

• Relationship types

- Person is colleague of Person
- Person is member of Team
- Person works on Project
- Team consists of Person
- Team has leader Professor
- Team cooperates with Team
- Team is part of Institution
- Project consists of Person who is a member of Team
Identification of Characteristics (Step 1.3)

• Example
  ▪ Try to identify characteristics of persons:
    Each person has a name and is identified by a personal number. A person can be called to their phone numbers. We need to know at least one phone number. We also need to send them emails.
Identification of Characteristics (Step 1.3)

- Example

Each person has a **name** and is identified by a **personal number**. A person can be called to their **phone numbers**. We need to know at least one phone number. We also need to send them **emails**.

- Person characteristics
  - Personal number
  - Name
  - One or more **phone numbers**
  - Email
Identification of Characteristics (Step 1.3)

- Example
  - Try to identify characteristics of memberships:
    
    We need to know when a person became a member of a project and when they finished their membership.
Identification of Characteristics (Step 1.3)

• Example
We need to know \textbf{when} a person \textbf{became} a member of a project and \textbf{when} they \textbf{finished} their membership.

• Identified membership characteristics
  - From
  - To
Schema Creation (Step 2)

• Step 2 of conceptual modeling
  ▪ Model identified types using a suitable conceptual data model (i.e., create conceptual data schema) and visualize it as a diagram
  ▪ You can use various tools for modeling, so-called Case Tools, e.g.,
    – Commercial: Enterprise Architect, IBM Rational Rose, ...
    – Academic: eXolutio
Modeling Language Selection (Step 2.1)

• Which model should we choose?
  ▪ There are several available languages, each associated with a well-established visualization in diagrams
  ▪ We will focus on...
    – Unified Modeling Language (UML) class diagrams
    – Entity-Relationship model (ER)
  ▪ There are also others...
    – Object Constraints Language (OCL)
    – Object-Role Model (ORM)
    – Web Ontology Language (OWL)
    – Predicate Logic, Description Logic (DL)
Conceptual Schema Creation (Step 2.2)

• How to create a schema in a given language?
  ▪ Express identified types of entities, relationships and their characteristics with constructs offered by the selected conceptual modeling language
    – UML: classes, associations, attributes
    – ER: entity types, relationship types, attributes
Entity-Relationship Model (ER)
Unified Modeling Language (UML)
ER and UML Modeling Languages

• ER
  ▪ Not standardized, various notations and extensions (e.g., ISA hierarchy)

• UML
  ▪ Family of models such as **class diagrams**, use case diagrams, state diagrams, ...
    - Standardized by the OMG (Object Management Group)

• Note that...
  ▪ ER is more oriented to data design, UML to code design
  ▪ Both ER and UML are used in practice, but UML has become more popular recently
  ▪ ER constructs were incorporated to new versions of UML as well
Types of Entities

Type of real-world entities

Persons, research teams and research projects.

<table>
<thead>
<tr>
<th>UML</th>
<th>ER</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class</strong></td>
<td><strong>Entity type</strong></td>
</tr>
<tr>
<td>Name</td>
<td>Name</td>
</tr>
</tbody>
</table>
Characteristics of Entities

Attributes of a type of real-world entities

A person is characterized by their personal number, name, optional email address and one or more phone numbers.

<table>
<thead>
<tr>
<th>UML</th>
<th>ER</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attribute of a class</strong></td>
<td><strong>Attribute of an entity type</strong></td>
</tr>
<tr>
<td>Name and cardinality</td>
<td>Name and cardinality</td>
</tr>
</tbody>
</table>
Types of Relationships

Type of a relationship between two real-world entities

A team has one or more members, a person can be a member of zero or more teams. A team has exactly one leader, a person can be a leader of zero or more teams.

**UML**

<table>
<thead>
<tr>
<th>Class</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>personNumber, name</td>
</tr>
<tr>
<td>Team</td>
<td>name, url</td>
</tr>
</tbody>
</table>

**Binary association**: name and two participants with names and cardinalities

**ER**

**Binary relationship type**: name and two participants with cardinalities
Cardinalities in Relationships

Person
- personNumber
- name

Team
- name
- url

+has_member 1..*
member

+is_member_of 0..*

+leads 0..*
leader

Relationship
member

Relationship
leader

SIRET
XRG

Martin Nečaský
Tomáš Skopal
Jaroslav Pokorný
Josef Novák

SIRET
XRG

INTLIB
Characteristics of Relationships

Attributes of a type of relationship between real-world entities

A person is a team member within a given time interval

**UML**

**Attribute of a binary association class**
Name and cardinality

**ER**

**Attribute of a relationship type**
Name and cardinality
Generalization / Specialization

Type of entities which is a specialization of another type

Each person has a personal number and name. A professor is a person which also has one or more phones and can lead teams. A student is a person which also has a date of study beginning.

**UML**

**Generalization**: specific association with no name, roles and cardinalities

**ER**

**ISA hierarchy**: specific relationship with no name and cardinalities
Generalization / Specialization

• Note that...
  ▪ Entity type can be as a source for multiple hierarchies
  ▪ Each entity type can have at most one generalization

• Additional constraints
  ▪ **Covering constraint**
    – Each entity must be of at least one specific type
      • I.e. each Person is a Professor or Student (or both)
  ▪ **Overlap constraint**
    – Each entity must be of at most one specific type
      • I.e. there is no Student that would be a Professor at the same time
Composite Attributes

Structured characteristics of real-world entity types

A person has one or two addresses comprising of a street, city and country.

No specific construct
Auxiliary class

Composite attribute: name, cardinality and sub-attributes
Recursive Relationships

Type of a relationship between entities of the same type

A person has zero or more colleagues.

**UML**
- Normal association with the same participants

**ER**
- Normal relationship type with the same participants
N-ary Relationships

Type of a relationship between more than just two entities

A person works on a project but only as a team member.

**UML**

**N-ary association**
Similar to a binary association but with three or more participants

**ER**

**N-ary relationship type**
Similar to a binary relationship type but with three or more participants
N-ary Relationships

- Note that...
  - N-ary relationships can also have attributes
  - UML allows us to use **more expressive cardinalities**
    - E.g. a given combination of a particular person and project is related to zero or more teams through the given association

```plaintext
Nary associations
Team
- name
Person
- personNumber
Project
- projectNumber
worker
0..*
0..1
1..*
0..*
0..*
0..*
```

![Diagram showing N-ary relationships between Person, Project, and Team with cardinalities](image)
N-ary Relationships

- Can n-ary relationships be replaced with binary?
  - Which projects does Jaroslav Pokorný work on as a member of the SIRET research group?
  - I.e. what is the difference between the following?
N-ary Relationships

• Can n-ary relationships be replaced with binary?
  ▪ Which projects does Jaroslav Pokorný work on as a member of the SIRET research group?
  ▪ I.e. what is the difference between the following?
N-ary Relationships

- Can n-ary relationships be replaced with binary?
  - Yes, but in a different way...
  - N-ary association = class + separate binary association for each of the original participants

\[ \text{class Nary associations} \]

Team
- name
Person
- personNumber
Project
- projectNumber
Worker
0..*
0..1
1..*
0..*
0..*
0..*

Team
- name
Project
- projectNumber
Worker
0..*
0..1
1..*
0..*
0..*
0..*
Identifiers

Full identification of real-world entities

A person is identified either by their personal number or by a combination of their first name and surname.

<table>
<thead>
<tr>
<th>UML</th>
<th>ER</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>Attribute or a group of attributes marked as an <strong>identifier</strong></td>
</tr>
</tbody>
</table>
Identifiers

Partial identification of real-world entities

A team is identified by a combination of its name and a name of its institution.

<table>
<thead>
<tr>
<th>UML</th>
<th>ER</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>Attribute or a group of attributes marked as a partial identifier</td>
</tr>
</tbody>
</table>
Identifiers

• Note that...
  - Each entity type must always be identifiable
    - At least by a set of all its attributes if not specified explicitly
  - Partial identifiers create identification dependencies
    - Only (1,1) cardinality is allowed (makes a sense)!

• Entity types
  - Strong entity type
    - ... has at least one (full) identifier
  - Weak entity type
    - ... has no (full) identifier, and so at least one partial identifier
    - ... is both existentially and identification dependent
Data Types

Data type of attributes

A person has a personal number which is an integer and name, email and phone which are all strings.

- **Note that...**
  - Set of available data types is not specified strictly
  - Data types are actually not very important at the conceptual layer

<table>
<thead>
<tr>
<th>UML</th>
<th>ER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute of a class may have a data type assigned</td>
<td>Attribute of entity type may have a data type assigned</td>
</tr>
</tbody>
</table>
Complete Sample UML Diagram

```
class Velký příklad
Person
  - personNumber
  - name
  - email [0..1]

Team
  - name
  - url

Project
  - projectNumber
  - startYear

Professor
  - phone [1..*]
  - www
  - email
  - dblp

Institution
  - name
  - number
  - www

Member
  - from
  - to

Position
  - title
  - from
  - to

Worker
  - from
  - to

+has_coleague 0..*
+is_coleague_of 0..*

+has_member 1..*
+leads
0..*
leader
+has_leader 1
+is_colleague_of
0..*
+has_colleague
0..*
worker

+belongs_to 0..*
+employer 0..*

+cooperates_with 0..*
+is_member_of 0..*
+has_member 1..*
+leads
0..*
leader
+has_leader 1
+is_colleague_of
0..*
+has_colleague
0..*
```

Diagram showing relationships between classes such as Person, Team, Professor, Institution, Member, Position, and Worker, with attributes like name, email, phone, and URLs, and relationships like has_coleague, is_coleague_of, has_member, leads, has_leader, and is_colleague_of.
Object Constraint Language (OCL)
Object Constraint Language

- **OCL**
  - Language for **formal specification of advanced integrity constraints**
  - Part of a UML standard
  - Motivation
    - Cardinalities are not enough!
  - Supports invariants, derived values, method pre-conditions and post-conditions, etc.
    - We shortly focus on **invariants**...
OCL Example 1

Each project must start after year 1990.

context p : Project inv p.startYear > 1990
Each team with more than 10 members must have a project and people working on the project.

class Velký příklad
Person
- personNumber
- name
- email [0..1]
Team
- name
- url
Project
- projectNumber
- startYear
Professor
- phone [1..*]
- www
- email
- dblp
Institution
- name
- number
- www
Member
- from
- to
+worker
+has_member 1..*
+leads
0..*
leader
+has_leader 1
+is_colleague_of
0..*
+has_colleague
0..*
Position
- title
- from
- to
+employee
+employer
0..*
0..*
+belongs_to
1
+is_member_of
0..1
+cooperates_with
0..*
0..*
+is_member_of
0..*
+cooperates_with
0..*

context Team inv

self.has_member->size() > 10 implies
self.worker->size() > 0
OCL Example 3

A person can work on a project only when they are a member of a team which solves the project.

context Person inv

self.is_member_of->includesAll(self.worker.Team)
OCL Example 4

A person is identified by their personal number.

context p1, p2 : Person inv

   p1.personNumber = p2.personNumber implies p1 = p2
A team leader must be an employee of the institution of the team.

```
context t : Team inv
t.belongs_to.employee->exists(p | p = t.has_leader)
```