

B0B36DBS, BD6B36DBS: Database Systems

<http://www.ksi.mff.cuni.cz/~svoboda/courses/192-B0B36DBS/>

Lecture 1

Conceptual Modeling

Martin Svoboda

martin.svoboda@fel.cvut.cz

18. 2. 2020

Czech Technical University in Prague, Faculty of Electrical Engineering

Lecture Outline

- **Introduction to database systems**
 - What is a database?
 - Basic terminology
- **Conceptual database modeling**
 - ER – Entity-Relationship Model
 - UML – Unified Modeling Language

Basic Terminology

- **Database (DB)**

- Logically organized **collection of related data**
 - Self-describing, metadata stored together with data
 - Data + schema + integrity constraints + ...

- **Database management system (DBMS)**

- **Software** system enabling **access to a database**
 - Provides mechanisms to ensure security, reliability, concurrency, integrity of stored data, ...

- **Database system**

- Information system
 - Database, DBMS, **hardware, people, processes, ...**

"everything around DB"

Motivation for Databases

- Why database systems?
 - **Data sharing and reusability**
 - Consistency, correctness, compactness...
 - 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, ...

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

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 who is relevant for our application
 - E.g. users, investors, owners, domain experts, etc.

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 **often needed**
 - Each schema describes a given database application (applications) from a **different point of view**
 - Even different conceptual models may be needed
 - We only focus on **conceptual data viewpoint**

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

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 when a person became a member of a project and when they finished their membership.

- Identified membership characteristics
 - **From**
 - **To**

Schema Creation (Step 2)

- Step 2 of conceptual modeling
 - **Model** the identified types and characteristics using a suitable **conceptual data model** (i.e. create a conceptual data schema) and **visualize it as a diagram**
 - Various modeling tools (so-called **Case Tools**) can be used, 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 using 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)
 - <http://www.omg.org/spec/UML/>

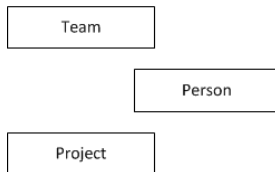
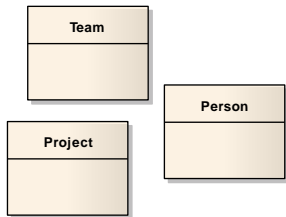
- **Note that...**

- 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**
- 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.



UML

ER

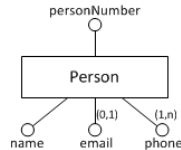
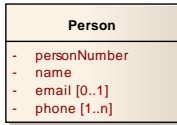
Class
Name

Entity type
Name

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.



UML

Attribute of a class
Name and cardinality

ER

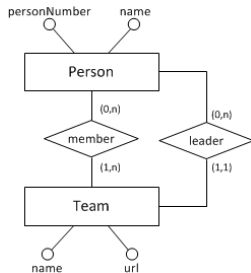
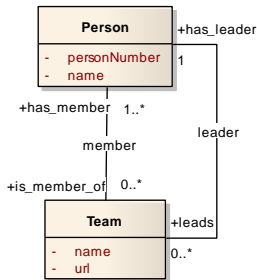
Attribute of an entity type
Name and cardinality

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.



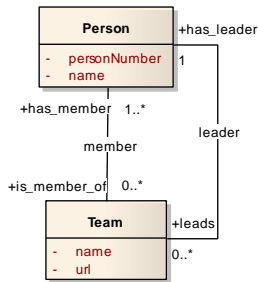
UML

ER

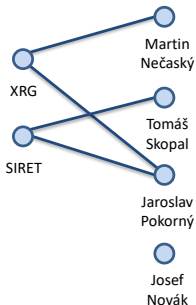
Binary association: name and two participants with names and cardinalities

Binary relationship type: name and two participants with cardinalities

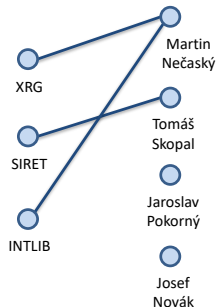
Cardinalities in Relationships



Relationship
member



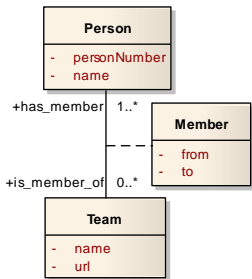
Relationship
leader



Characteristics of Relationships

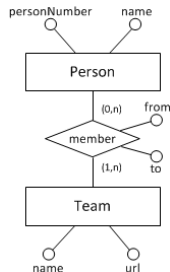
Attributes of a type of relationship between real-world entities

A person is a team member within a given time interval



UML

Attribute of a binary association class
Name and cardinality



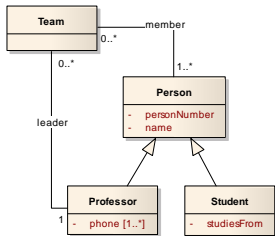
ER

Attribute of a relationship type
Name and cardinality

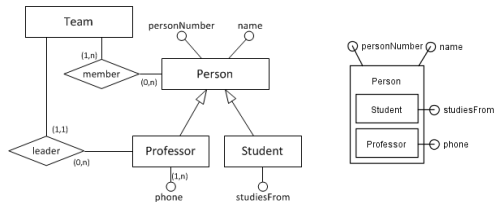
Generalization / Specialization

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



ER

Generalization: specific association with no name, roles and cardinalities

ISA hierarchy: specific relationship with no name and cardinalities

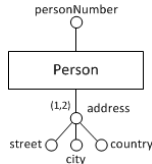
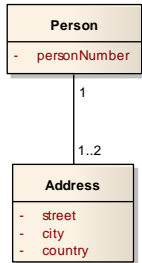
Generalization / Specialization

- Note that...
 - Entity type can be a source for multiple hierarchies
 - Each entity type can have at most one generalization
- Additional constraints
 - **Covering constraint** (complete/partial)
 - Each entity must be of at least one specific type
 - I.e. each Person is a Professor or Student (or both)
 - **Disjointness constraint** (exclusive/overlapping)
 - 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.



UML

No specific construct
Auxiliary **class**

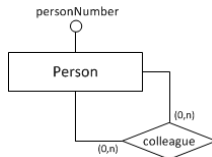
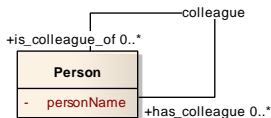
ER

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

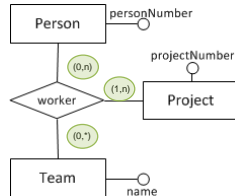
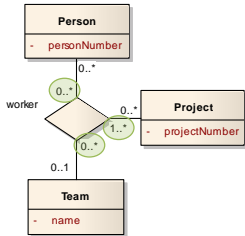
ER

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

ER

N-ary association

Similar to a binary association but with three or more participants

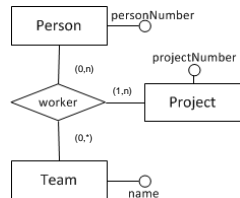
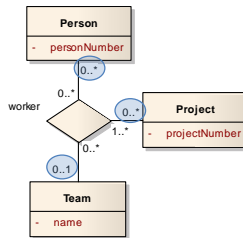
N-ary relationship type

Similar to a binary relationship type but with three or more participants

N-ary Relationships

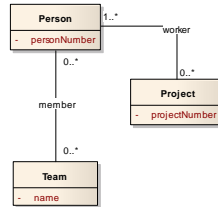
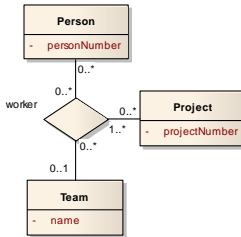
- Note that...
 - N-ary relationships can also have attributes
 - UML allows us to use **more expressive cardinalities**
 - E.g. a given combination of a particular person and project is related to zero or more teams through the given association

— ...



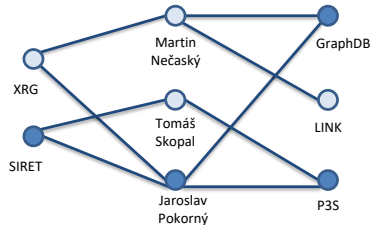
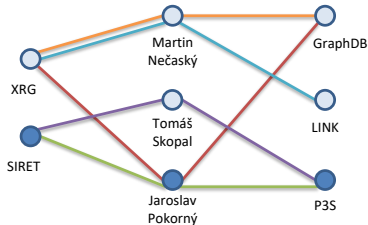
N-ary Relationships

- 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?



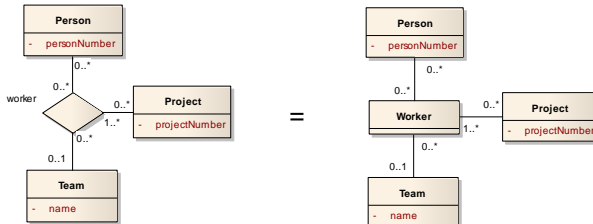
N-ary Relationships

- 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

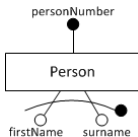
- **Can n-ary relationships be replaced with binary?**
 - Yes, but in a different way...
 - N-ary association = class + separate binary association for each of the original participants



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.

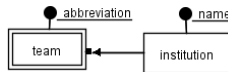
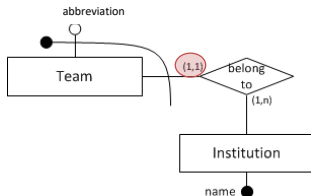


UML	ER
N/A	Attribute or a group of attributes marked as an identifier

Identifiers

Partial identification of real-world entities

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



UML

ER

N/A

Attribute or a group of attributes marked as a partial **identifier**

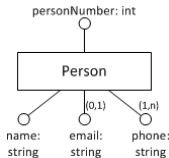
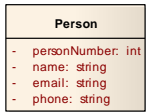
Identifiers

- 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.



UML

Attribute of a class may have a data type assigned

ER

Attribute of entity type may have a data type assigned

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

Sample UML Diagram

