Lecture 1

Introduction to Databases. Fundamental Concepts.

Databases

- **Grading / Final Grade**
  - In the examination period time (retake examination period time).
  - Written Exam (W) – 50%
  - Practical Test (P) – 25%
  - Lab Grade (L) – 25%
  - *** Requirements to pass: Written Exam>=5 AND Practical Test>=5**
  - W – in examination period time
  - P – in week 13 / week 14
- E-mail: emilia@cs.ubbcluj.ro, emiliapop_23@yahoo.com
- Webpage: http://www.cs.ubbcluj.ro/~emilia
- Each student must have at least 5 seminar attendances and at least 6 laboratory attendances - to attend the exam in the examination period / retake examination period, according to “the Computer Science Department’s Decision” https://www.cs.ubbcluj.ro/wp-content/uploads/Hotarare-CDI-29.04.2020.pdf
- Lab requirements: 4 lab assignments
  - No more than 2 lab assignments can be delivered / lab.
  - Lab assignment delay = 1p / lab (=2 weeks).
  - Lab assignments cannot be delivered in week 13, week 14 and examination period.
  - In retake examination period can be delivered a maximum number of 2 lab assignments, with a penalty of 35% (maximum grade is 6.5 / lab assignment), only if the practical test is retaken (exception, for the students that took 10 on the practical test).

Background

- Databases – in virtual way exist everywhere: education, social networks, research, media, financial services, e-commerce, tourism, ….
- A lot of complex **data sets** are used nowadays.
- The data owners (individuals, organisations, companies, …) need to ***efficiently manage*** *their data, to exact the correct information in a proper time.*
- *So, to handle the data management are need powerful and flexible* ***Data Management Systems.***

Databases

- Database = a huge collection of data kept for a long period of time to be analyzed, managed and used later.
- Manage different aspects of the real world or some concepts, by using a ***data model***.
- A database is useful to *store and manage data*.
- A database is available on any computer / laptop and can be access easily.
- Almost every day a database is used. (e.g. search for a phone number, pay a product directly or by card, transfer in a bank account)
- Databases are not a concept of a computer.
- Databases out of computer: phone book, dictionars, …

Databases

- An application has:
  - Data stored in *files* or *databases* (or *distributed databases*)
  - A management algorithm
  - An user interface
- ***Files***
- e.g. a tourism agency stores a large collection of data related to the employees, clients, transactions, …; requirements:
  - Quick answers related to the data
  - Protecting the data from inconsistent changes done by users that access data in the same time
  - Restricting access to some parts of data (e.g. salaries)

Databases

- ***Files*** – should be used for single-user program and for a small amount of data
- When store and manage data by using a collection of files, some difficulties may arrise:
    - Data redundacy (parts of data can be stored in multiple files -> possible inconsistencies)
    - Different data storage format
    - Read / write operations from program can create difficulties in program development (changes in structures may cause changes in program)
    - Modifications to the data (update, delete) may increase the process of retrieving data on the search criteria (difficult operations)
    - Integrity constraints – are checked in the program
    - Main memory mangement (e.g. a huge amount of data should be retrieve)
    - Weak security policies (allows different users to access different segments of data)
    - The management of concurrent data is difficult
    - Data has to be restored in a consistent state when a system fails (e.g. a transaction not successfully finished should return the payment to the client)

Data Models

- *Data Model* = a collection of concepts (used to define the structure and the syntax of the data, the consistency constraints of data and of relationships between the data) used to describe data.
- Data should be described accordingly to a model (to be managed automatically)
- *Schema of the database = Data structures* used to describe the components of a data collection stored in a database
  - The structure is defined usualy in the beginning of the developments process of the software application.
- *Instance of the database* = data from a defined structure (e.g. type variables from the language programs)
- *Instance of the schema* = data in the collection (e.g. classes and objects in OOP)

Data Models

- *Data Model – Schema –* ***Instance of a database*** – Examples:

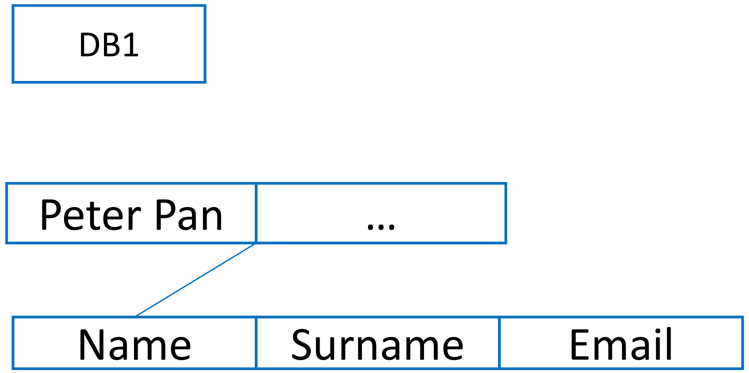
- Data Model is a *set of records*. The records may contain PersonId, Name, Surname, Address, ListOfCourses, Photo.
- Data Model is a *XML Document.* The document may contain the list of the books with the identical id as an identifier, title, author name/names (as sub-elements).
- Data Model is a *graph*. The nodes can represent cities and the arches between the nodes can represent high ways.

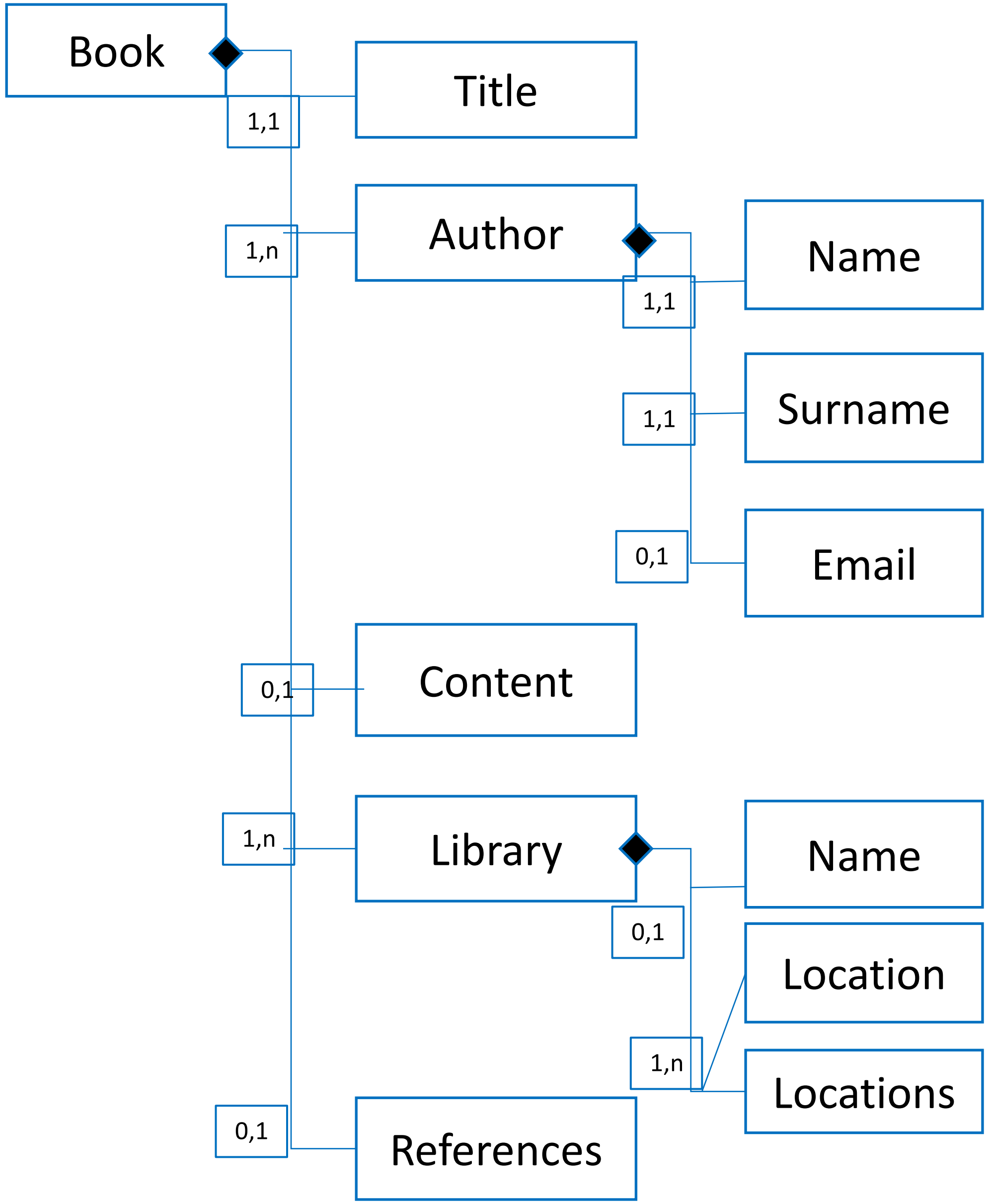
Data Models

- Entity-Relationship (1990s)
- Relational (1970s) (1NF)
- Network (1965)
- Hierarchical (1965)
- Object-Oriented
- noSQL
- Semistructured (XML) (1990s)

Data Models

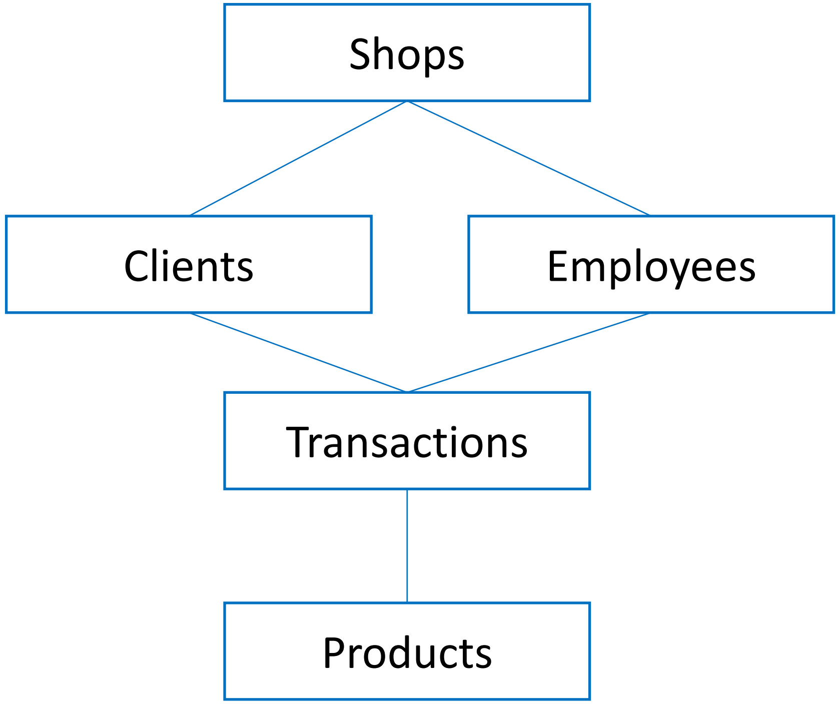
- ***Hierarchical Data Model***
  - The first data model (from ‘60s)
  - Represent an extension of a store / processing system files.
  - Organize the data in a tree structure.
  - E.g. The structure of an entity called *Book* in the Hierarchical Data Model and 1 instance





Data Models

- ***Network Data Model***
  - *It is an extension of the Hierarchical Data Model*
  - *Organize the data in a graph structure*
  - *E.g. (in the right side)*



- ***Object-Oriented Model***
  - Introduce new concepts (e.g. class, attribute, method) and relationships between them (e.g. association, aggregation, inheritance)
  - Popular for analyze, projection and soft development
  - In Databases it is just “scientific” due to its efficiency

Data Models

- ***Relational Model***
  - In ‘70s Ted Codd invents this model and the concept of abstract data.
  - The most popular model nowadays (the one that we will use in this course).

- **Relation** – is the main concept used to describe data
- The schema of a relation has:
  - name for the relation
  - for each field (column) : name and type

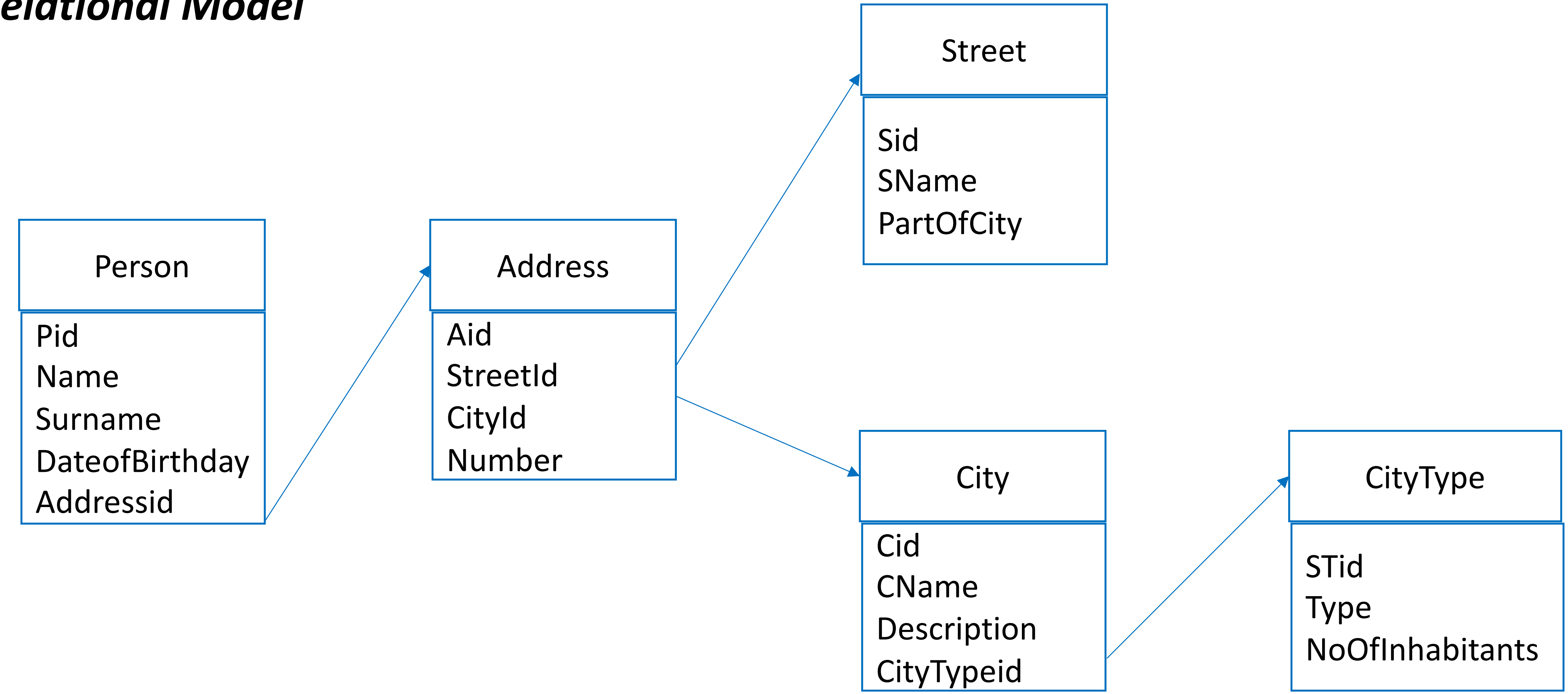
E.g. Person (Pid: integer, Name: string, Surname: string, DateOfBirthday: date)

- An instance of the Person relation
- Every row has 4 columns

Pid	Name	Surname	DateOfBirthday
11	John	Smith	02/05/1980
13	Pamela	Cobb	07/03/2000
24	Katie	Burrow	11/06/1993
26	Flint	Hught	08/12/2001
45	Daniel	Cabe	05/09/1989

Data Models

- ***Relational Model***



Person

Pid
Name
Surname
DateofBirthday
Addressid

Address

Aid
StreetId
CityId
Number

Street

Sid
SName
PartOfCity

City

Cid
CName
Description
CityTypeid

CityType

STid
Type
NoOfInhabitants

Data Models

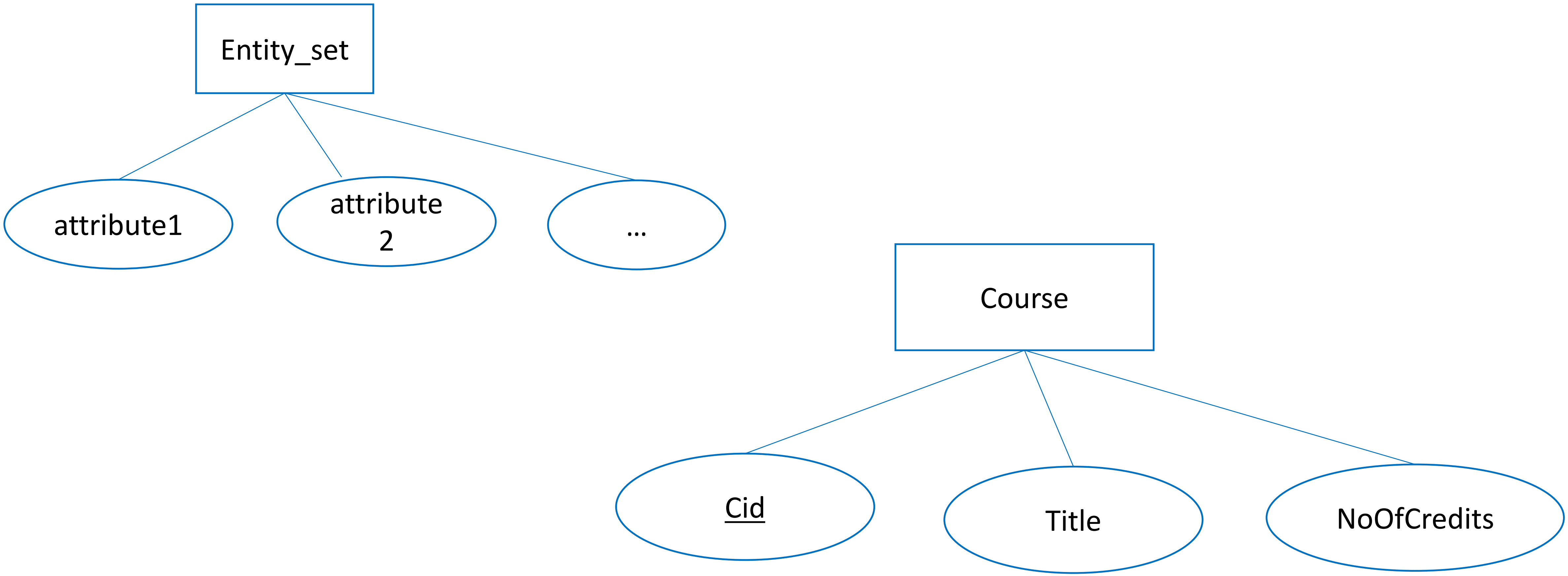
- ***Entity-Relationship Model***
  - Abstract, semantic, high-level model
  - Closer to the manner in which the data is stored than to the user's perspective on the data
  - Helps in a good initial description of the data
  - The design is presented in terms of the DBMSs (Database Management System's) model
    - E.g. the Entity-Relationship Model corresponds to a Relational mapping
  - main concepts used: entities, attributes, relationships.
  - **Entity** – an object from the real world, a piece of data described by properties (attributes)
  - **Entity set** - the entity with the schema – the entity name and the list of attributes (e.g. the set of persons)
  - **Attribute** – has name, domain for values (type), conditions to check correctness
  - **Key** – a restriction defined on an entity set; set of attributes with distinct values in the entity set's instances

Data Models

- ***Entity-Relationship Model***
  - **Relationship** – describe a relation (association) between 2 or more entities; can include descriptive attributes
  - **Relationship set** (Relationship schema) – include all the relationships with the same structure (name, entity sets used in relation, descriptive attributes)
  - **Schema of the model** – contains the entity sets and the relationship sets
- **Binary relationships** - are established between 2 sets (A and B) and determine the following *relationship types:*
  - **1:1** – one A entity can be associated with just one B entity and one B entity can be associated with just one A entity
    - E.g. Group - LeaderGroup
  - **1:n** - one A entity can be associated with multiple B entities and one B entity can be associated with just one A entity
    - E.g. Group - Student
  - **m:n -** one A entity can be associated with multiple B entities and one B entity can be associated with multiple A entities
    - E.g. Student – Courses
      - The changes of the database are checked in the specific of each relationship (restrictions)

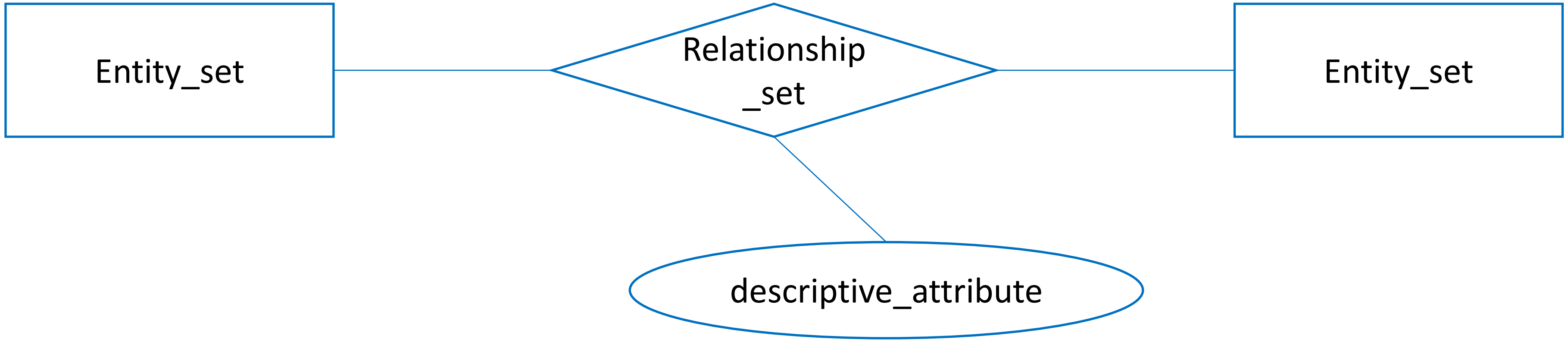
Data Models

- ***Entity-Relationship Model*** - representation of an entity set and associated attributes

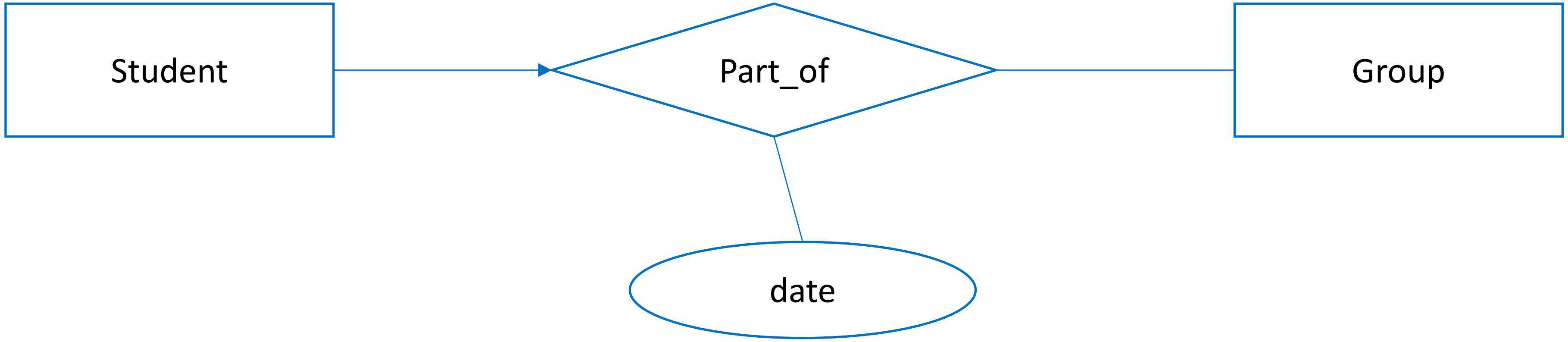


Data Models

- ***Entity-Relationship Model*** - representation of a relationship set with associated attributes

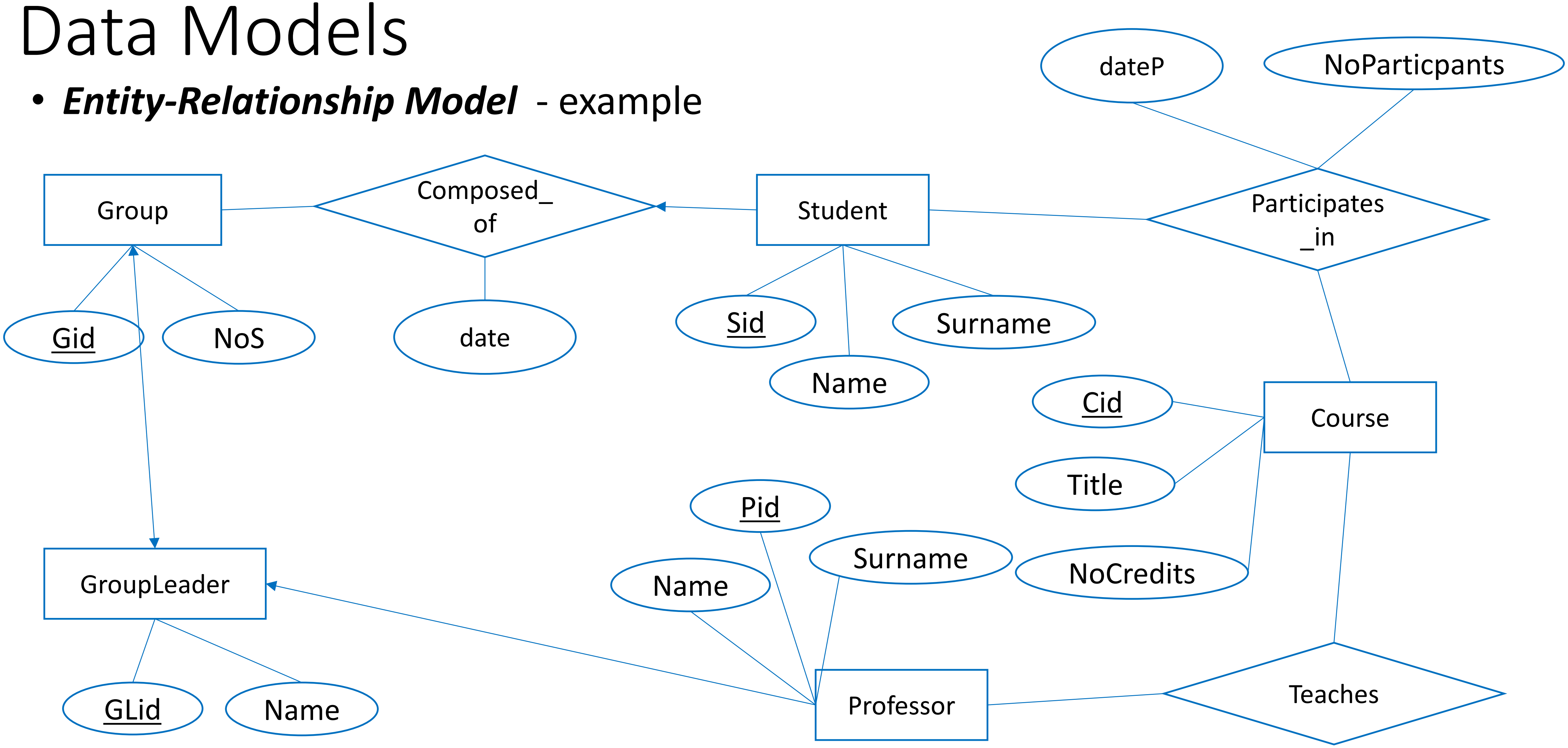


1:n relationship set



Data Models

- ***Entity-Relationship Model*** - example



Databases & DBMS

- A **database** contain:
  - The *database schema* - a *description of data structure* (used to model the data) and kept in the database dictionary
  - A *collection of data* - instances of the schema
  - *Components*: stored procedures, functions, views, users, …
- Database **design**
  - The organization in terms of the data in a database
- Database **analysis**
  - Extracting information that involves the data from the database by using queries
- **Data Definition** - in the database dictionary
- **Data Management** (insert/update/delete) and querying
- **DBMS – Database Management System** – set of programms used to manage a database
- DBMS examples: SQL Server, Oracle, DB2 (IBM), Informix (IBM), Teradata, MySQL, PostgreSQL, Access, Foxpro, …
- **Database system** – database and DBMS

Databases & DBMS

- **Database** = structured collection of data
- **DBMS** = applications that allow the user to extract and manipulate data inside of a database
- DBMS – a collection of instruments that
  - Create a database and mantain it correspondingly
  - Querying and modify correctly the data from the database
  - Secure data
  - Allow the shared access to multiple users
- Use databases when:
  - Need persistency
  - Big quantity of data
  - Structured data
  - Need concurrent and distributed acces to the data
  - Need integrity and security
  - Need to share data with other applications

- Don’t use databases when:
  - The investment is too big
  - Need a lot of effort
  - We develop a simple and well-defined application that doesn’t need future changes
  - No need to share the access to the data to multiple users

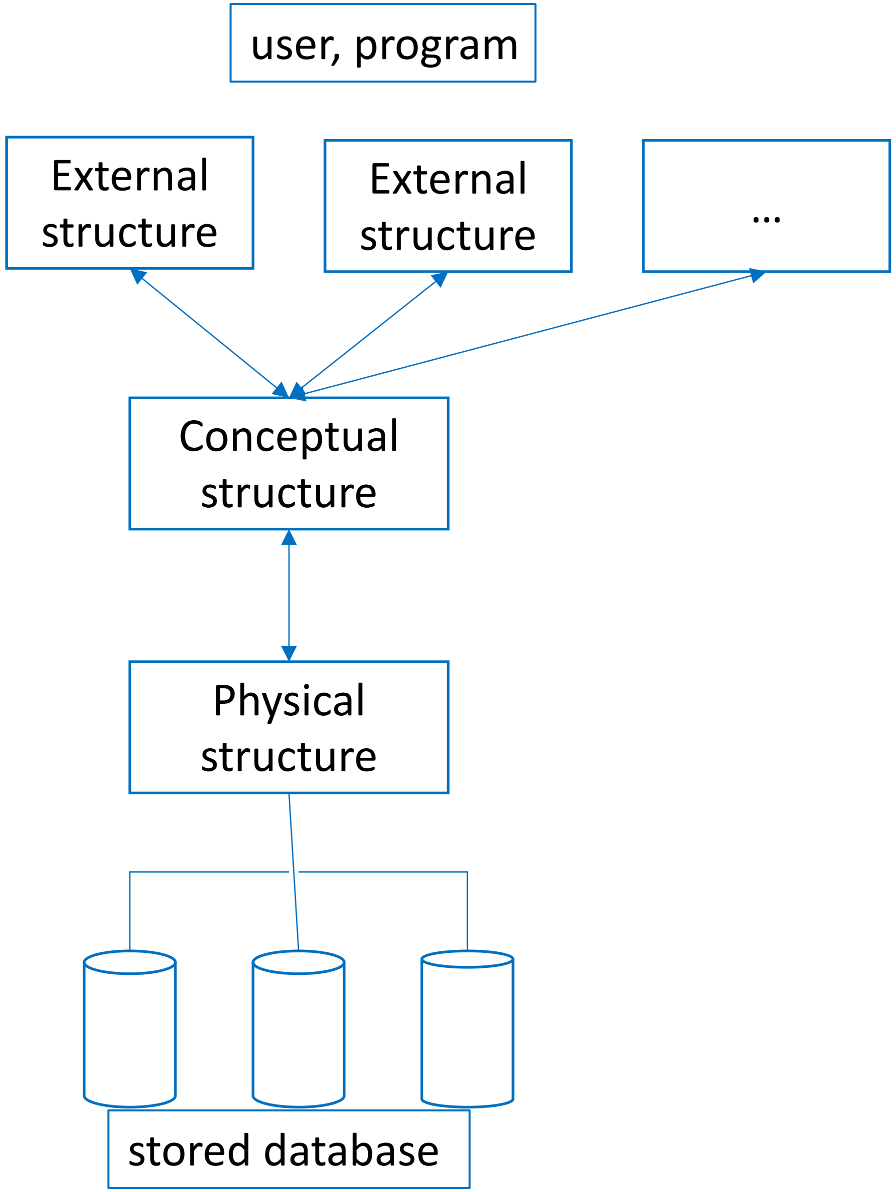
Solution: text files.

Data versus Structure

- **Structure of the database**
  - Usualy it is not modified
  - Called *metadata* (=data that represent data)
- **Database state**
  - Modified frequently
  - DBMS (Database Management System) ensure the validaty of the state of the database
- **Instance of the database**
  - Usualy refers to the combination between the structure and the state of the database

Structure of a Database

- How to organize and store information about a subject in a database such that the users could operate with entities and relationships inside it
- DBMS stores data in a form that need a large number of bits
- The levels of abstractization in DBMS help to treat the differences between how a database is designed and how it is implemented and stored
- **ANSI-SPARC architecture** – proposed in 1975 with 3 level architecture for a database system and include
    - ***Conceptual structure (database schema)*** - the data structure and the restrictions in the database
    - ***External structures*** – data structures used by an user / program
    - ***Physical structure (internal structure)*** – storage structures in the database (e.g. indexes, data files, …)



Structure of a Database

- **Examples:**
- ***Conceptual structure (database schema)*** - information about entities and relationships between them
  - e.g. Student participates to Course
  - Student (Sid:int, Name:string, Surname:string)
  - Course (Cid:string, Title:string, NoOfCredits:int)
  - Participation (Sid:int, Cid:string, Pdata:date)
- ***External structures*** – can be several external structures customized to each category of users
  - e.g. external structure with all the Student participations to the Course
  - BestParticipation (Sid:int, Cid:string, NoOfParticipations:int) – adding it to the conceptual schema, causes redundancy, the database may become full of errors.
- ***Physical structure (internal structure)*** – information related to how the relations are stored on the disk, creation of indexes (=data structure that speed up the queries)
  - e.g. the relations are stored as unsorted files of records; the indexes are created on the first column of Students and Courses relations

Logical independence, Physical independence

- **Data independence** – due to the 3 levels of abstraction, the applications can be isolated from the changes in the data structure / storage
- **Logical data independence** – the programs that use the data from the database are not affected by changes in the conceptual structure
  - Applications can be developed in several stages
  - e.g. Course relation can be replaced by:
    - MandatoryCourse (Cid:string, Title:string, NoOfCredits:in)
    - OptionalCourse (Cid:string, Title:string)
- **Physical independence** – the applications are not affected from the changes in physical state of the data
  - Files can be added to improve the optimization process (e.g. indexes); users' programs does not check the files directly (the physical structure)

References:

- C.J. Date, *An Introduction to Database Systems (8th Edition)*, Addison-Wesley, 2003.
- H. Garcia-Molina, J. Ullman, J. Widom, *Database Systems: The Complete Book, Prentice Hall Press*, 2008.
- G. Hansen, J. Hansen, *Database Management And Design (2nd Edition),* Prentice Hall, 1996.
- R. Ramakrishnan, J. Gehrke, *Database Management Systems*, McGraw- Hill, 2007. http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed.html
- R. Ramakrishnan, J. Gehrke, *Database Management Systems (2nd Edition),* McGraw-Hill, 2000.
- A. Silberschatz, H. Korth, S. Sudarshan, *Database System Concepts*, McGraw-Hill, 2010. http://codex.cs.yale.edu/avi/db-book/
- L. Țâmbulea, *Curs Baze de date*, Facultatea de Matematică și Informatică, UBB, 2013-2014.
- J. Ullman, J. Widom, *A First Course in Database Systems*, http://infolab.stanford.edu/~ullman/fcdb.html