Databases

Lecture 1

Introduction to Databases. Fundamental Concepts

Databases

- Lecture2 + Seminar1 + Laboratory2
- grading
  - written exam (W) - 50%
  - practical exam (P) - 25%
  - labs (L) - 25%
  - W, P >= 5
- To attend the exam, a student must have at least 12 laboratory attendances and at least 5 seminar attendances, according to the Computer Science Department's decision: http://www.cs.ubbcluj.ro/wp-content/uploads/Hotarare-CDI-15.03.2017.pdf
- http://www.cs.ubbcluj.ro/~sabina
- sabina@cs.ubbcluj.ro

Context

- databases - virtually everywhere
  - education
  - research
  - financial services
  - media
  - e-commerce
  - social networks
  - tourism
  - telecommunications
  - …
- e.g., personal expenses management, analyzing security breach data set

Context

- unprecedented data growth, huge & complex **data sets**
- data owners (organizations, individuals, etc) need to **efficiently manage** their data, to extract correct information in a timely manner, so as to support their own operations & interests

=> the need for powerful & flexible **data management systems** that simplify data management

1. The Components of an Application

- data (stored in files or databases)
- management algorithm
- user interface

2. Data Storage Methods

- files
- databases
- distributed databases

3. Files: Characteristics and Limitations

* ex: bank storing a large collection of data on employees, clients, accounts, transactions, etc; requirements:
    - quick answers to questions about the data
    - protecting the data from inconsistent changes made by users accessing it at the same time
    - restricting access to some parts of the data, e.g., salaries
- difficulties encountered when storing and managing the data using a collection of files ->

3. Files: Characteristics and Limitations

- multiple data storage formats
- data redundancy
  - some parts of the data can be stored in multiple files => potential inconsistencies
- read / write operations are described in the program; certain record structures are taken into account => difficulties in program development (when the file structure changes, the program needs to be changed as well)
- changing data (modifying / removing records), retrieving data based on search criteria - difficult operations
- integrity constraints are checked in the program
- main memory management, e.g., how is a data collection of tens / hundreds of GB loaded for processing?

3. Files: Characteristics and Limitations

- no adequate security policies
  - the context calls for security policies that allow different users to access different portions of data
- concurrent data access is difficult to manage
- data must be restored to a consistent state in the event of a system failure, e.g., a bank transaction transferring money from account A to account B is interrupted by a blackout after having debited account A, but prior to crediting account B; the money must be put back in account A
- files are useful for single-user programs that require a small amount of data

4. Data Description Models

- data must be described according to a model (to be managed automatically)
- a **data description model** is a set of **concepts and rules** used to model the data; such concepts describe:
  - the structure of the data;
  - the consistency constraints;
  - the relationships with other data.
- when using a model, in order to describe a collection of data stored in a database, certain data structures are used; these constitute the **schema** of the database (the data structure or template); the data in the collection follows the schema and can be considered an **instance** of the schema (analogy: classes and objects in object-oriented programming)
- the data description constructs in a model are *high-level*, hiding many low-level details about data storage, e.g., there's a long way from the *Student* entity to the computer stored bits ☺

4. Data Description Models

- entity-relationship
- relational
- network
- hierarchical
- object-oriented
- noSQL
- semistructured (XML)
- data streams

4. Data Description Models: The Relational Model

- the main concept used to describe data is the **relation**, i.e., a set of records
- the schema of a relation specifies the name of the relation, the name and type of each field (column)
  - e.g., representing information about movies:

    Movie(*mid*: string, *title*: string, *director*: string, *year*: integer)

4. Data Description Models: The Relational Model

- e.g., instance of the Movie relation; every record has 4 fields:

mid	*title*	*director*	*year*
84386	Hibernatus	Édouard Molinaro	1969
7583	Moscow Does Not Believe in Tears	Vladimir Menshov	1980
47288	Close Encounters of the Third Kind	Steven Spielberg	1977
32	Contact	Robert Zemeckis	1997
46747	E.T. the Extra-Terrestrial	Steven Spielberg	1982

4. Data Description Models: The Entity-Relationship Model

- a **semantic**, more abstract, high-level model
- it significantly eases the task of developing a good initial description of the data
- such a semantic model is useful since, even though the database management system's model hides many details, it's still closer to the manner in which the data is stored than to the user's perspective on the data
- a design in such a model is subsequently expressed in terms of the database management system's model
  - e.g., the ER -> relational mapping

4. Data Description Models: The Entity-Relationship Model

- the main **concepts** used in this model are: entities, attributes, relationships
- the **entity**
  - is a piece of data, an object in the real world
  - it's specified through certain attributes (properties)
- entities with the same structure (e.g., the set of students) are instances of an **entity set** (class / entity schema)
- an entity set has a name and a list of attributes
- when specifying an entity set, an **attribute** has: a name, a domain of possible values and possible conditions to check whether the values are correct
- a **key** can be defined on an entity set: a set of attributes with distinct values in the entity set's instances; such a key is a restriction

4. Data Description Models: The Entity-Relationship Model

- **the relationship** – is a piece of data; it specifies an association among 2 or more entities; additional attributes can be used
- all relationships with the same structure (involving entities of the same types) are described by a **relationship set** (relationship schema)
- a relationship set has a name, the entity sets used in the association and possible descriptive attributes

- **the schema of the model**
  - a set of entity sets and relationship sets

4. Data Description Models: The Entity-Relationship Model

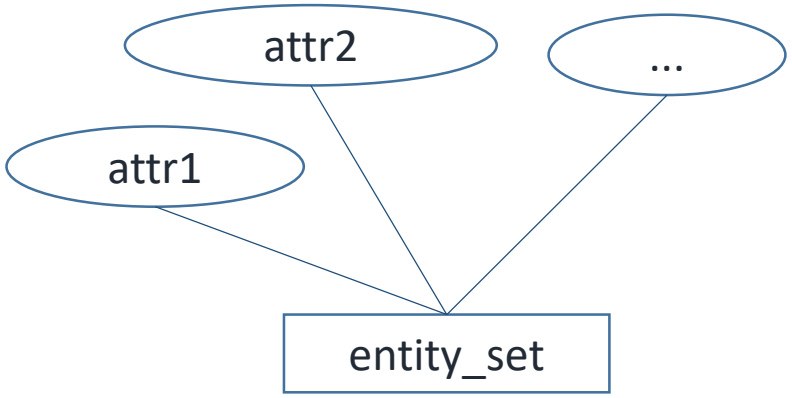
- for **binary relationships** (involving entity sets T1 and T2), the following **relationship types** can be defined:
  - **1:1**: one T1 entity can be associated with at most one T2 entity, and one T2 entity can be associated with at most one T1 entity
    - e.g., the association between group and faculty member (e.g., to specify the groups' tutors)
  - **1:n**: one T1 entity can be associated with any number of T2 entities, and one T2 entity can be associated with at most one T1 entity
    - e.g., the association between group and students
  - **m:n:** one T1 entity can be associated with any number of T2 entities, and one T2 entity can be associated with any number of T1 entities
    - e.g., the association between courses and students

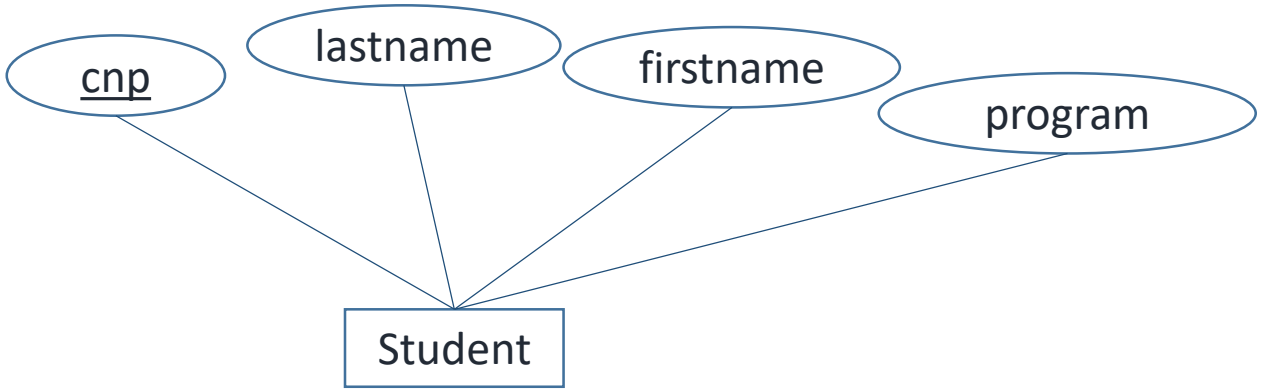
4. Data Description Models: The Entity-Relationship Model

- these can be considered as **restrictions** in the database; every time the database is changed, the system must check whether the relationship is of the specified type

4. Data Description Models: The Entity-Relationship Model

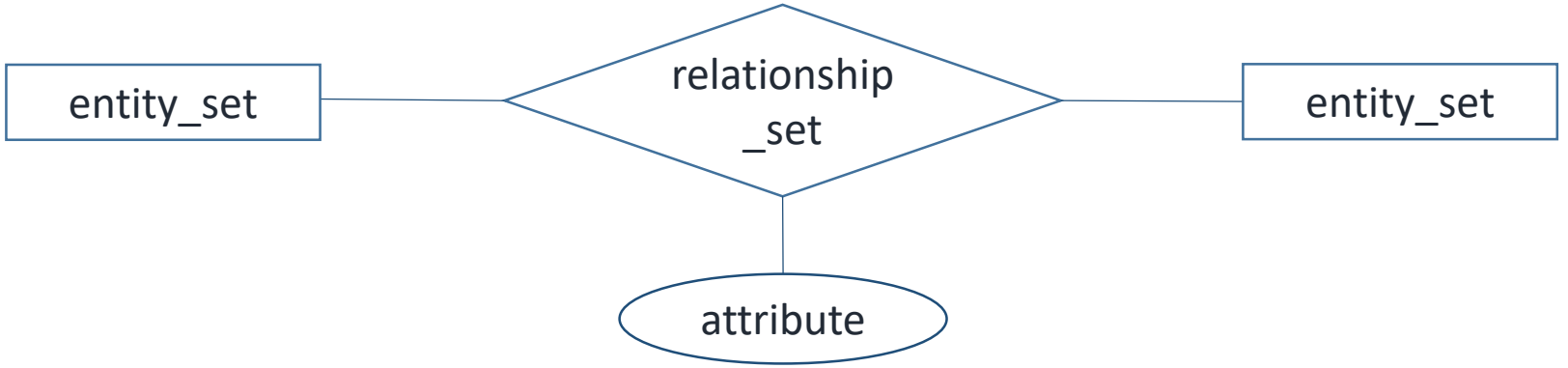
- graphical representation of the model
  - entity set and associated attributes

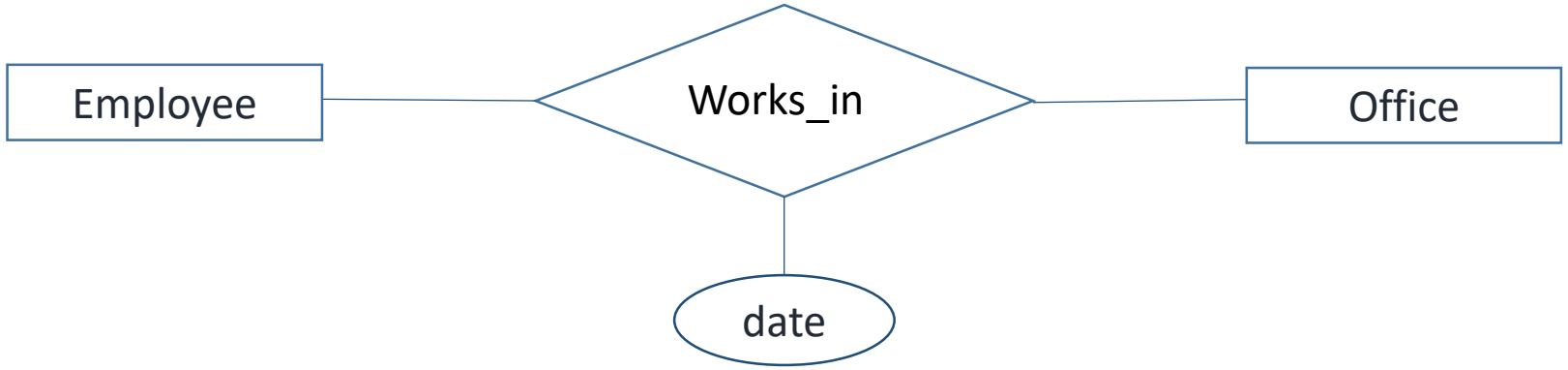




4. Data Description Models: The Entity-Relationship Model

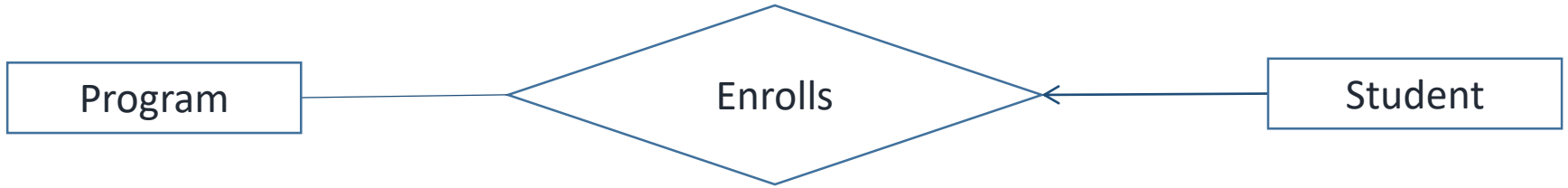
- graphical representation of the model
  - relationship set and associated attributes





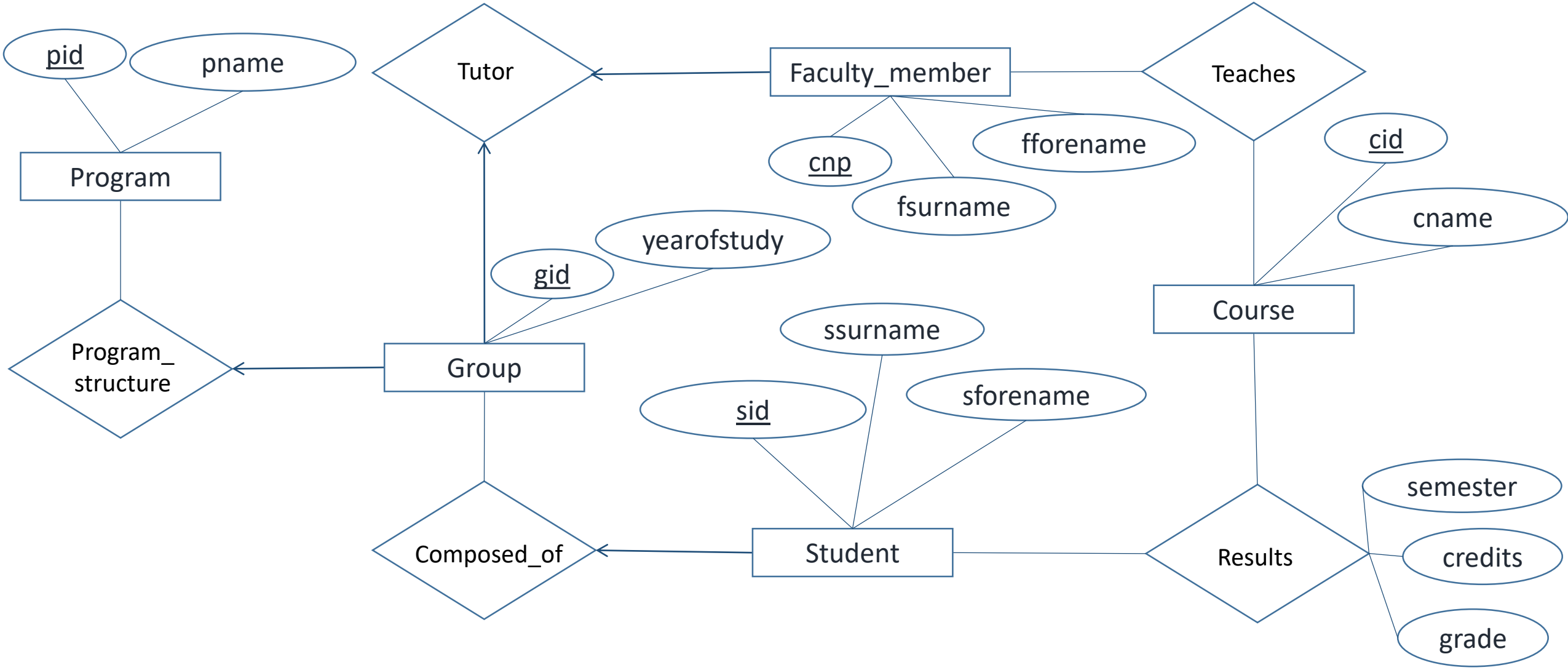
4. Data Description Models: The Entity-Relationship Model

- graphical representation of the model
  - 1:n relationship sets - graphical convention:



4. Data Description Models: The Entity-Relationship Model

- example:



5. Databases and Database Management Systems

- a **database** contains:
  - **the description of the data structures** used to model the data (the database schema) in a database dictionary*
  - a **collection of data** - instances of the schema
  - various **components**: views, procedures, functions, roles, users, etc
- separation between:
  - **data definition** (kept in the database dictionary)
  - **data management** (insert / delete / update) and querying
- database **design**
  - describe an organization in terms of the data in a database
- data **analysis**
  - answer questions about the organization by formulating queries that involve the data in the database

* dictionary (catalog) - contains metadata; a possible schema:

(attr_name, relation_name, attr_type, attr_position_in_relation)

5. Databases and DBMSs

- **database management system** (DBMS)
  - set of programs that manage a database
- DBMS examples
  - Oracle, DB2 (IBM), SQL Server, Sybase (SAP), Informix (IBM), Teradata, MySQL, PostgreSQL, Access, Paradox, Foxpro, ...
- **database system**
  - database + DBMS

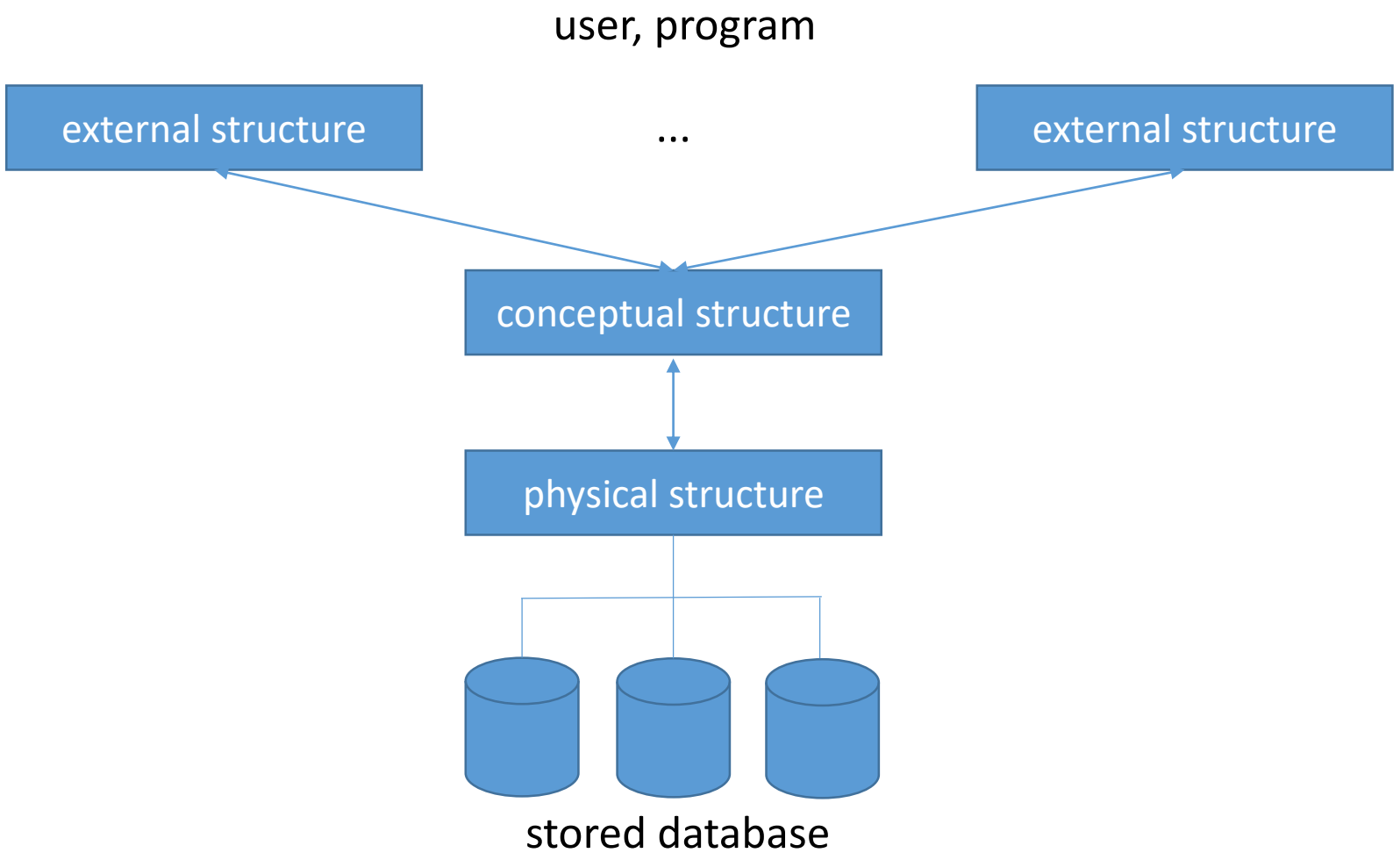
6. The Structures of a Database

- when thinking about how to organize and store information about an organization in a database, users operate with high-level concepts corresponding to the entities and relationships in the organization
- on the other hand, the DBMS stores data in the form of a very large number of bits
- the difference between the way users think about their data and the manner in which the data is stored is reconciled through the levels of abstraction in a DBMS

6. The Structures of a Database

- the **ANSI-SPARC architecture** - a three-level architecture for a database system, proposed in 1975; in general, this model is used by the main management systems and includes:
  - **the conceptual structure (the database schema)**: it describes all the data structures and restrictions used in the database
  - **external structures**: describe the data structures used by a particular user / program; the description employs a certain model, and the DBMS can find the data in the conceptual structure
  - **the physical structure (internal structure)**: describes the storage structures for the database (data files, indexes, etc)

6. The Structures of a Database



6. The Structures of a Database

- e.g., the conceptual structure
  - information about entities, e.g., students, courses, and relationships among entities, e.g., grades earned by students:

    Student(*sid*: string, *slastname*: string, *sfirstname*: string, *gpa*: real)

    Teacher(*tid*: string, *tlastname*: string, *tfirstname*: string, *salary*: real)

    Course(*cid*: string, *cname*: string)

    Grade(*sid*: string, *cid*: string, *grade*: real)

    Teaches(*tid*: string, *cid*: string)

6. The Structures of a Database

- e.g., the physical structure
  - information about how relations are stored on the disk, about the creation of indexes (data structures that speed up data retrieval operations):

    relations – stored as unsorted files of records

    indexes are created on the first column of the Student and Teacher relations

6. The Structures of a Database

- e.g., external structure with information about the best result for each student (the student's sid, last name and first name, the name of the course and the corresponding grade)

BestResults(*sid*: string, *slastname*: string, *sfirstname*: string, *cname*: string, *grade*: real)

- BestResults is a *view*, i.e., conceptually, it's a relation, but its records are not stored in the system; instead, they are computed on demand using the view's definition, which takes into account the relations stored in the database
- introducing BestResults in the conceptual schema => redundancy, the database is prone to errors, e.g., a record for a new grade is introduced in the Grade relation without operating a corresponding (necessary) change in the BestResults relation
- a database can contain multiple external structures, each being customized for a certain group of users

7. Logical Independence and Physical Independence

- **data independence**: due to the 3 levels of abstraction, applications are insulated from changes in the data structure / storage
- **logical data independence**: changes in the conceptual structure don't affect programs using data from the database
  - important: applications can be developed in several stages
    - e.g., the Student relation is replaced by:

      StudentPublic(*sid*: string, *slastname*: string, *sfirstname*: string)
      StudentPrivate(*sid*: string, *gpa*: real, *dob*: date)

  the definition of BestResults can be changed to take into account StudentPublic; a user querying BestResults will get the same result as before the conceptual schema was changed

7. Logical Independence and Physical Independence

- **physical data independence**: applications are not affected by changes in the physical structure
  - important: files (e.g., index) can be added for optimization purposes; users' programs don't check the files (the physical structure) directly

8. Functions of Database Management Systems

- database **definition**: definition language (or dedicated applications that generate commands in the definition language)
- data **management**: insert, update, delete, querying
- database **administration**: database access authorization, database usage monitoring, database performance monitoring, database performance optimization, etc
- the **protection** of the database: *confidentiality* (protection against unauthorized data access), *integrity* (protection against inconsistent changes)

9. Types of Database Users

- database **administrators**
- database **designers**
  - schemas, restrictions, functions and procedures available to users, necessary optimizations
- data management **application users**
- **application programmers**

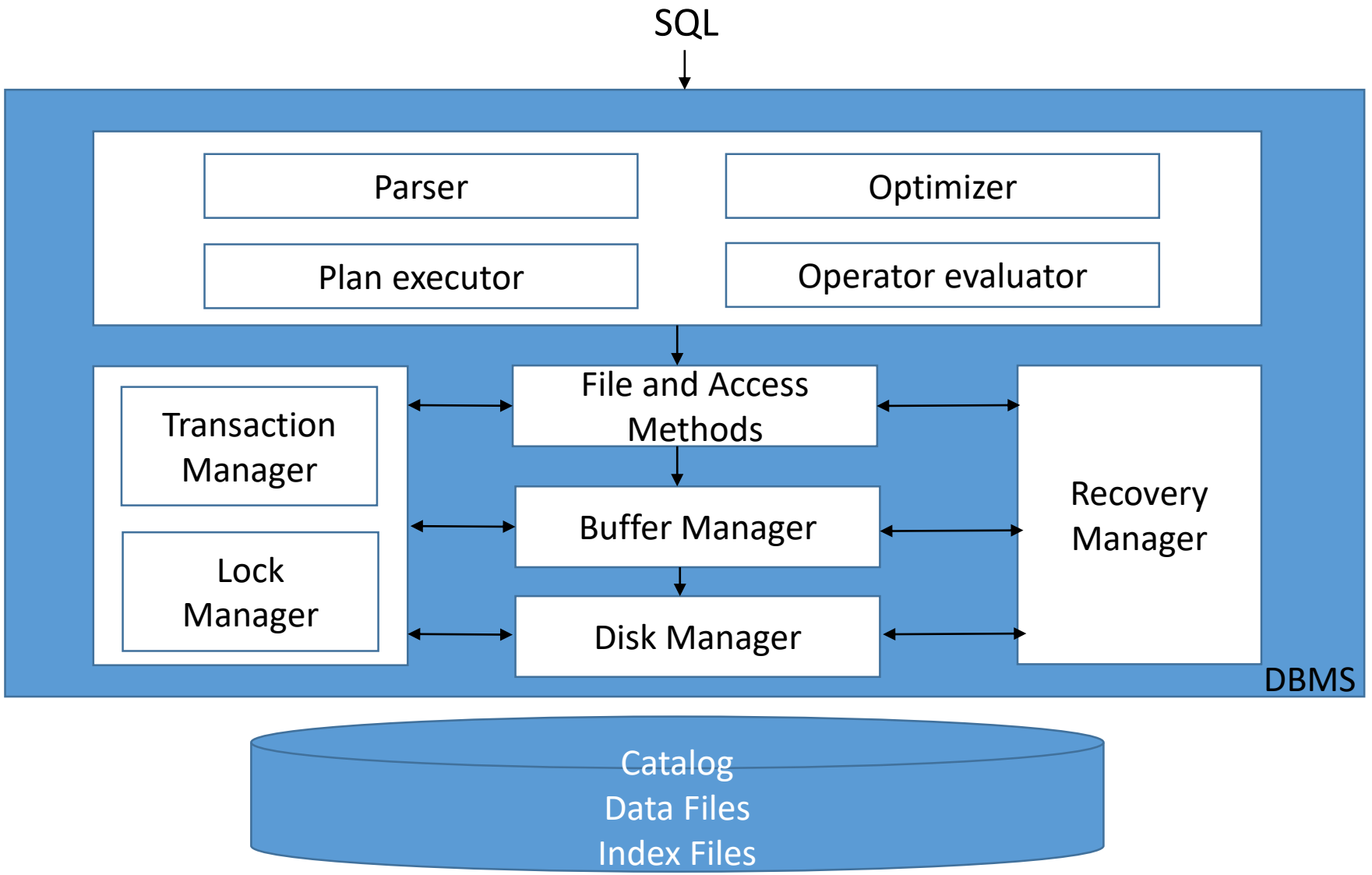
9. Types of Database Users

* applications are developed in various languages / programming environments (Web, Java, .NET apps, etc)

- to execute an operation on the database, the app sends a command to the database system; the system executes the command and sends the answer back to the app (client / server technology)
- the request is written in SQL (Structured Query Language), a standard language for data access

10. The Architecture of a DBMS

- [Ra07] proposes the following structure for a DBMS:



10. The Architecture of a DBMS

- SQL commands can come from different user interfaces (Web Forms, SQL interface, etc), and can be included in applications written in various programming languages, e.g., Java, C#, etc
- Optimizer
  - produces an efficient execution plan for query evaluation, taking into account storage information
- File & Access Methods, Buffer Manager, Disk Manager
  - abstraction of files, bringing pages from the disk into memory, managing disk space
- Transaction Manager, Lock Manager
  - concurrency control, monitoring lock requests, granting locks when database objects become available

10. The Architecture of a DBMS

- Recovery Manager
  - recovery after a crash

11. Using a DBMS - Advantages

- a DBMS can manage large collections of interrelated data
- applications are not managing database implementation details: they send a SQL command and receive an answer (the DBMS evaluates the command, using various data access programs)
- it allows systems to be developed in several stages (changing the database schema, changing the applications, developing new applications)
- it optimizes data access (extremely useful for very large collections; it uses sophisticated techniques to store and retrieve data efficiently)
- data can be accessed from applications developed in a variety of languages / programming environments

11. Using a DBMS - Advantages

- if data is always accessed through the system, it is up to date and correct (integrity constraints are automatically checked)
- database access control (for users with different roles)
- concurrent access management
- it enables data recovery (log)
- data can be imported / exported using various formats
- it provides data analysis tools (data mining)
- it reduces application development time

References

- [Ta13] ȚÂMBULEA, L., Curs Baze de date, Facultatea de Matematică și Informatică, UBB, 2013-2014
- [Ra00] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems (2nd Edition), McGraw-Hill, 2000
- [Da03] DATE, C.J., An Introduction to Database Systems (8th Edition), Addison-Wesley, 2003
- [Ga08] GARCIA-MOLINA, H., ULLMAN, J., WIDOM, J., Database Systems: The Complete Book, Prentice Hall Press, 2008
- [Ha96] HANSEN, G., HANSEN, J., Database Management And Design (2nd Edition), Prentice Hall, 1996
- [Ra07] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems, McGraw-Hill, 2007, http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed.html
- [Si10] SILBERSCHATZ, A., KORTH, H., SUDARSHAN, S., Database System Concepts, McGraw-Hill, 2010, http://codex.cs.yale.edu/avi/db-book/
- [Ul11] ULLMAN, J., WIDOM, J., A First Course in Database Systems, http://infolab.stanford.edu/~ullman/fcdb.html