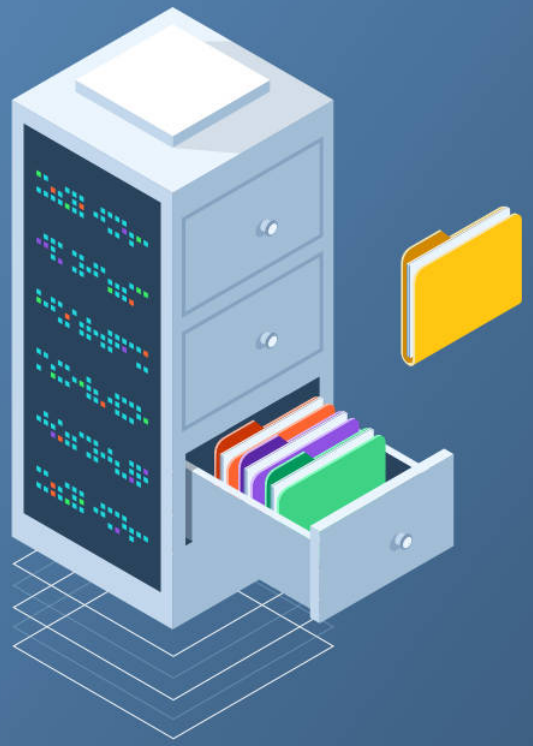
Introduction to databases and database management systems

The content of the slides is authored by Prof. Maria de Marsico and by Prof. Marina Moscarini, Sapienza University of Rome



Module topics

We will cover the following main topics

Relational algebra

$\sigma, \pi, \rho, \times, \bowtie, \cap, \cup, \ldots$

Concurrency

Some stuff about concurrency

Database design

How to design a good database?
Third normal form (3NF)

Physical organization

How does a database store data on the disk?

Structured information

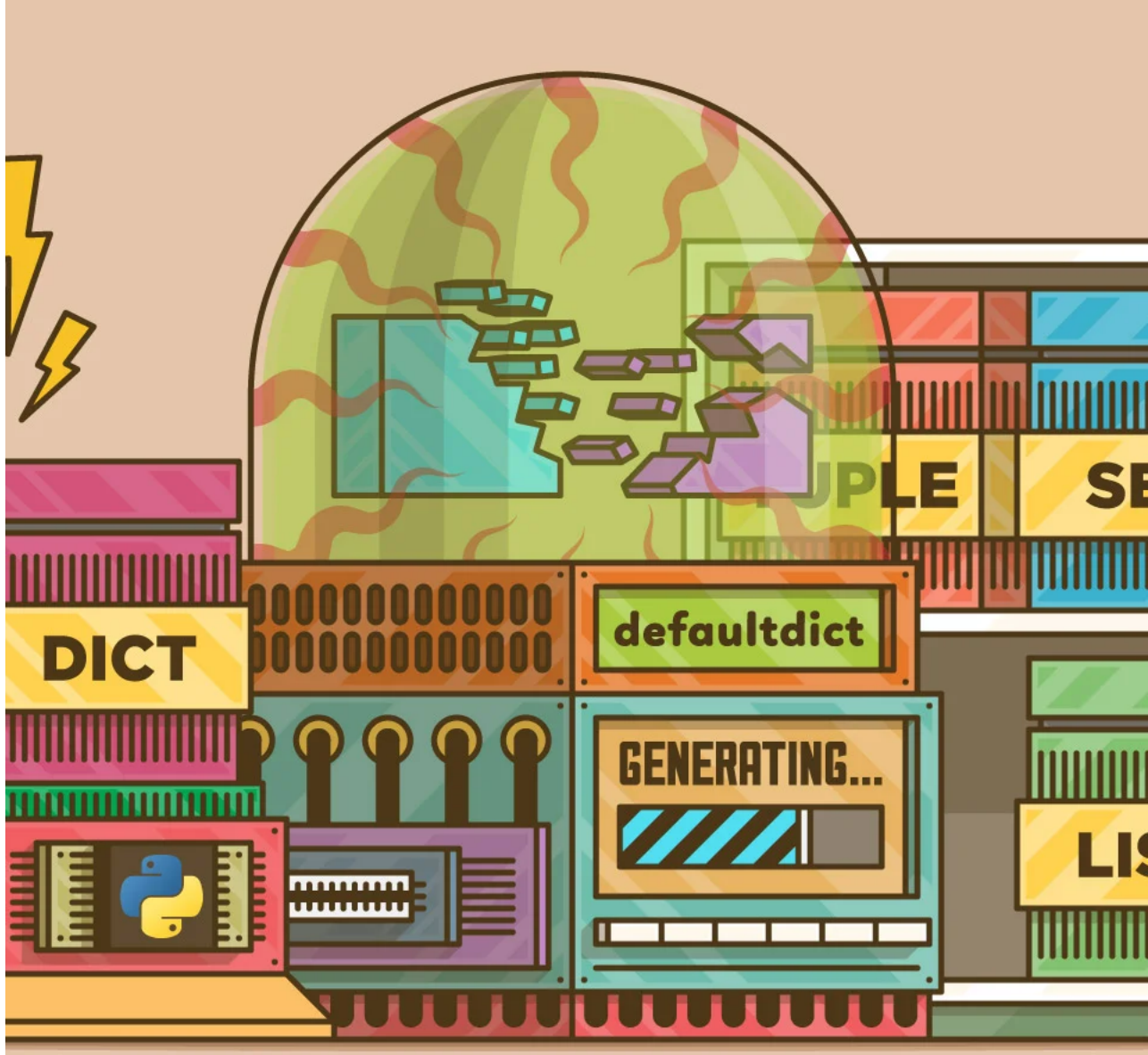
Information in electronic format can be recorded as

Structured data

Objects are represented by short strings of symbols and numbers

Unstructured data

Texts written in a natural language



Information System

An Information System is a component of an organization that is used to manage *(acquire, process, store, communicate)* information

Normally, the Information System operates in support of the other components of the organization

The notion of Information System is independent of its computerization

What happened earlier?

Each application had its own private file

File: sequential organization

Application: written in a file management oriented language *(Cobol, PL/1)*

Data management: file system

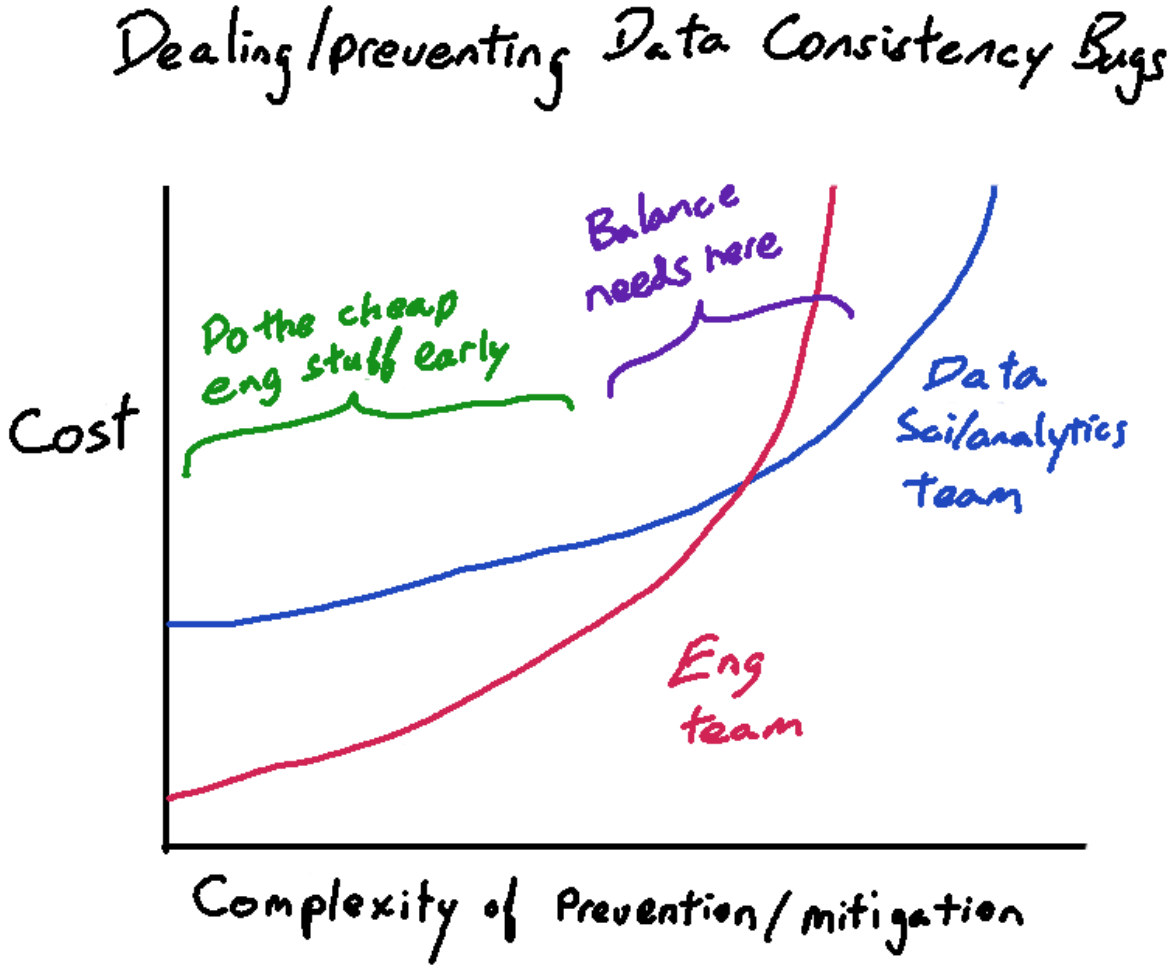
```
   05 DATABASE-KEY PIC X(10).
   05 DATABASE-VALUE PIC X(20).

FD OUTPUT-FILE.
01 OUTPUT-LINE PIC X(30).

WORKING-STORAGE SECTION.
01 END-OF-FILE PIC X VALUE 'N'.
01 DATABASE-COUNTER PIC 9(3) VALUE 0.

PROCEDURE DIVISION.
MAIN-PARAGRAPH.
   OPEN INPUT DATABASE-FILE
   OPEN OUTPUT OUTPUT-FILE

   READ DATABASE-FILE
      AT END MOVE 'Y' TO END-OF-FILE
   END-READ

   PERFORM UNTIL END-OF-FILE = 'Y'
      ADD 1 TO DATABASE-COUNTER

      MOVE DATABASE-RECORD TO OUTPUT-LINE
      WRITE OUTPUT-LINE

      READ DATABASE-FILE
         AT END MOVE 'Y' TO END-OF-FILE
      END-READ
END-PERFORM
```

Disadvantages

Redundancy: if two applications used the same data, it was replicated

Inconsistency: the update of a data item could relate to a single copy of the data

Data dependency: each application organized its data according to its intended use



DBMS

A **Database** (DB) is a set of **mutually linked files**

Data is organized in different **data structures** that facilitate their creation, access and updating and optimize the management of physical resources

The **Database Management System** (DBMS) is a software tool for the management of large amounts *(structured, processable, shared)* of data

Components of an information system

Database (DB)

Database Management System (DBMS)

Application software

Computer hardware (e.g., storage devices)

Personnel developing, managing or using the system

Storage

Database systems use **files** to store data but they provide users with an **abstract view** of the data, so that storage and manipulation details are transparent to them

Structured information

The structure of the information depends on its use and can be changed over time. To store data about a **person** over time we could use:

- first and last name (until a few centuries ago, this was not obvious either)
- name, surname, date of birth and place of birth
- tax number

etc...

Processing information

Goal

To facilitate the **processing of data** based on its properties

Structured data

Individual access to the elements of the structure is possible through **queries** *(interrogations)*, to retrieve information or perform calculations. Relationships between individual data items are represented in the **record structure**

Shared information

In an **organization**, each component is interested in a **portion** of the Information System

These portions may overlap:

- a database is an integrated resource **shared** by several components
- integration and sharing allow to **reduce redundancies** *(partially or totally replicated data)* **and** consequent **inconsistencies**

Shared information

Database sharing is never complete:

- control **privacy** and **access** regulation
- database sharing implies the need to manage simultaneous access to the same data: control of **concurrency**

Information System

An Information System is a set of **data** physically organized in **secondary memory** and managed in such a way as to allow its *creation, updating and interrogation*

Information System

Data is conceptually organized in **aggregates of homogeneous information** that constitute the components of the information system, and each update operation is targeted to a single aggregate, while a query may involve *one or more than one aggregate*

In databases:

- **files**: aggregates of homogeneous information
- **indexes**: files that allow you to quickly retrieve information from the "*main files*"

Data vs information

In computer systems, information is represented in the form of **data**: raw facts that need to be interpreted and correlated to provide information

Example: "*Maurizio Mancini*" and *0649255161* are a **string** and a **number**, i.e., two pieces of data

If they are returned in response to the **question** "who is the course instructor and what is their telephone number" then they constitute **information**

Data Model

The Data Model is the set of structures used to organize the **data** of interest and their **relationships**

Essential component: type constructors

For example, the **relational model** provides the relation builder: it organizes data as a set of homogeneous **records** *(types)*

Two main types of models

Logical models

Independent of physical structures but available in DBMSs: e.g. network, hierarchical, relational, object-oriented

Conceptual models

Independent from the modalities of realization, they have the scope to represent the entities of the real world and their relations in the first phases of the planning: e.g., **Entity-Relationship**

Historical notes (TODO: 17)

hierarchical mid-60s: first systems Generalized Update Access Method (IBM, Project Apollo, 1964) DL/1 (Data Language 1) (IBM, on the market in 1966) IMS (Information Management System) netted I-D-S (Integrated Data Store) (General Electric) CODASYL / DBTG / network systems

Mesh model

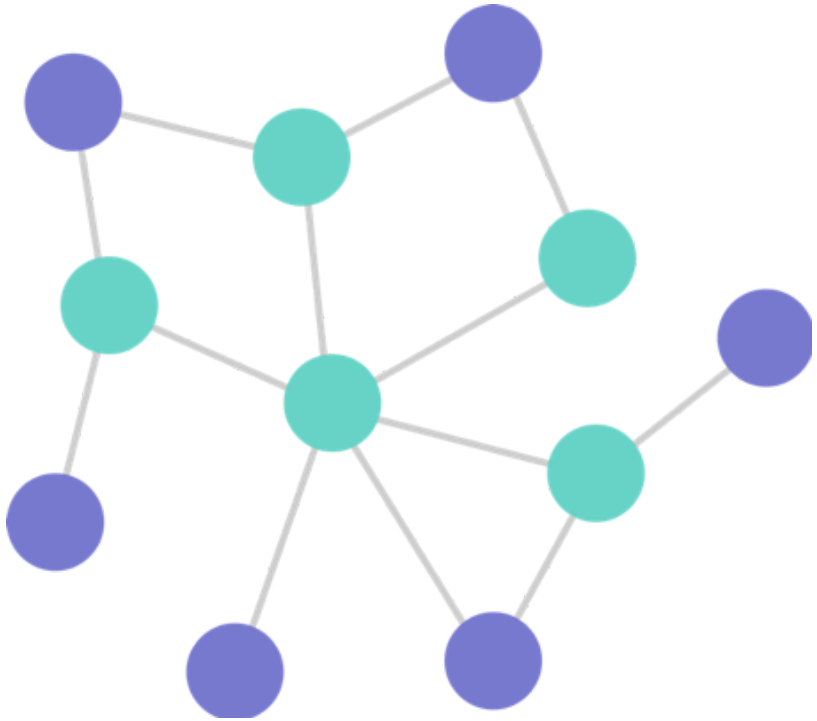
The data is represented as a collection of **records** of homogeneous type

Binary relationships are represented as links *(implemented as pointers = dependence on the physical structure of the database)*

The model is represented as a graph structure where:

- **nodes** are records
- **arches** are links

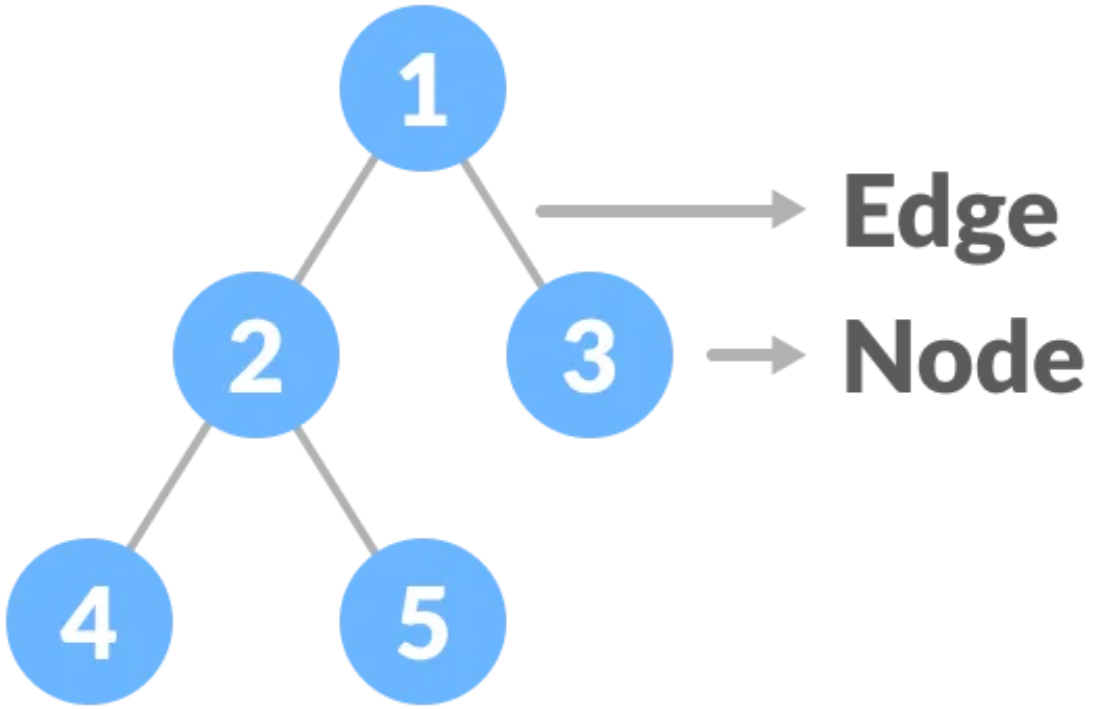
The most popular mesh model is **CODASYL**



Hierarchical model

It's a **restricted** type of mesh model:

- **hierarchy** = mesh composed of a collection of trees *(forest)*
- each **node** has only one parent



Historical notes

1970: E.F. Codd (IBM) introduces the relational model

70s: relational systems prototypes (System R, IBM)

80s: commercial relational systems (Ingress, Oracle, ...)

Relational model

Data and **relationships** are represented as **values**

There are no explicit references, i.e. pointers as in the mesh and hierarchical models $\Longrightarrow$ higher level representation

Relational model

An **object** is a **record**, **fields/attributes** are information of interest

Example: "*staff member*" is an object, and "*code, surname, first name, role, hiring year*" are information of interest about the "*staff member*"

CODE	SURNAME	NAME	ROLE	HIRING YEAR
COD1	Rossi	Mario	Analyst	1995

Relational model

A **table** is *set of records* of homogeneous type

Example: "*staff table*" is a set of records of type "*staff member*"

CODE	SURNAME	NAME	ROLE	HIRING YEAR
COD1	Rossi	Mario	Analyst	1995
COD2	Bianchi	Peter	Analyst	1990
COD3	Neri	Paolo	Admin	1985

Example of relational DB

STUDENTS

Matric	Last name	Name	Birthday
276545	Smith	Mary	25/11/1980
485745	Black	Anna	23/04/1981
200768	Greens	Paolo	12/02/1981
587614	Smith	Lucy	10/10/1980
937653	Brown	Mavis	01/12/1980

COURSES

Code	Title	Tutor
01	Physics	Grant
03	Chemistry	Beale
04	Chemistry	Clark

EXAMS

Stud	Vote	Course
276545	C	01
276545	B	04
937653	B	01

Example of mesh DB

> TODO: todo (slide 26)

Historical notes

> TODO: use a timeline, maybe with mermaid

- mid-80s: first object-oriented systems (O2, initially INRIA and later O2 Technology)
- starting from '93: definition of a standard (Object Data Management Group)

Object model

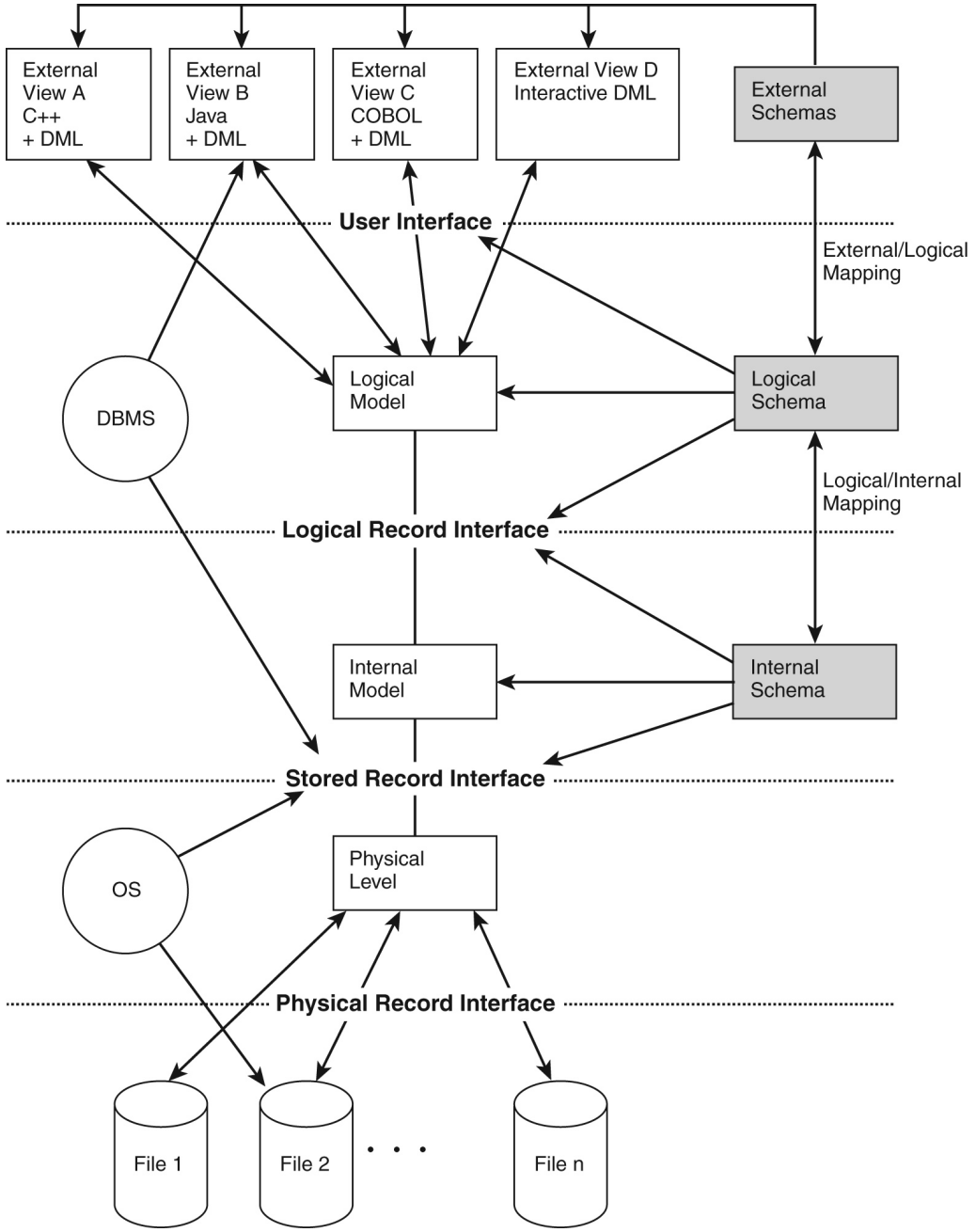
Model based on **objects**, **classes**, etc...

- **attributes** describe the state of an object
- **methods** (actions) describe the behavior of an object
- **objects** encapsulate both states and behaviors

There is no universally recognised model yet

The three abstraction levels of a DB

This course focuses on the the **logical schema**



The three abstraction levels of a DB

External schema

A description of a **portion** of the DB in a **logical model** through partial (derived) "views". It may provide **different data** organizations than those used in the logical schema and that reflect the needs and access privileges of particular types of users; **more than one external schema** may be associated with a logical schema

Logical schema

A description of the entire database in the "main" logical model of the DBMS, e.g., table structure

Physical schema

A representation of the logical schema by means of physical storage structures, i.e., files

A view (external schema)

logical schema
COURSES

Course	Lecturer	Room
DBs	Mancini	DS1
Systems	Rossi	N3
Networks	Bianchi	N3
Controls	Bruni	G

logical schema
CLASSROOMS

Name	Building	Floor
DS1	IMO	G
N3	IMO	G
G	Math	1

external schema (view)
COURSES LOCATIONS

Course	Room	Building	Floor
Systems	N3	IMO	G
Networks	N3	IMO	
Controls	G	Math	1

Data access

Database access occurs **only** through the **external schema**, which may coincide completely with the logical schema

Physical independence

The logical and external levels are independent from the physical one

A **relation is used in the same way** whatever its physical realization *(organization of files and their physical allocation)*

A the **physical implementation can change** without having to change the programmes

Logical independence

The **external level is independent** of the logical level

Additions or changes to views do not require changes to the logical layer

Changes to the logic schema that leave the external schema unchanged **are transparent**

Schemes and instances

In every database there exist

The schema

Substantially invariant in time, that describes its structure (intentional aspect): in the relational model, table headers are a list of attribute names and their types

The instance

The current values, which can change also very quickly (extensional aspect): in the relational model, the "body" of each table

Schemes and instances

> TODO: highlight schema and instance in this table

NAME	SURNAME	BIRTH	TOWN
Piero	Naples	22-10-63	Bari
Marco	Bianchi	01-05-54	Rome
Maria	Rossi	09-02-68	Milan
Maria	Bianchi	07-12-70	Bari
Paolo	Sossi	15-03-75	Palermo

Database languages

Data definition language (DDL)

For the definition of schemes (logical, external, physical) and other general operations

Data manipulation language (DML)

For querying and updating (instances of) databases

Structured Query Language (SQL)

SQL is a standardized language for databases based on the relational model (RDBMS). In SQL the two types of functionality are integrated into a single command language

Databases summary

Features

- multipurpose
- integration
- independence of data
- centralised control
  (DBA: database administrator)

Advantages

- minimum redundancy
- independence of data
- integrity
- security

Integrity

The data must satisfy "**constraints**" that exist in the context of interest

- a student resides in only one city *(functional dependencies)*
- the matriculation number uniquely identifies a student *(key constraints)*
- a grade is a positive integer between 18 and 30 *(domain constraints)*
- the overtime of an employee is given by the product of the number of hours and the hourly wage
- the salary of an employee cannot decrease *(dynamic constraints)*

Security

Data must be protected from unauthorized access

The DBA must consider:

- current value of the information for the organization
- who can access what data and in what way and then decide:
  - access regulation
  - effects of a violation

Data must be protected from hardware and software malfunctions and from concurrent access to the database

DB Transactions

Transaction

A sequence of operations constituting a single logical transaction

"Transfer €1000 from account c1 to account c2"

1. search c1
2. change balance to balance-1000
3. search c2
4. change balance to balance+1000

A transaction must be **fully executed** (committed) or not executed at all (rolled back)

Restore

To restore a correct database value we can use:

- a **transaction log** (contains transaction details: values before and after the change)
- a **dump** (periodic copy of the database)

Competition

Transaction 1: "Credit euro 1000 to c/c c1", **Transaction 2:** "Credit euro 500 to c/c c1"

> NOTE!!! Once a value has been read, each transaction modifies it in its own memory space

Transaction 1	Weather	Transaction 2
search c1	t1	
	t2	search c1
change balance to balance+1000	t3	
	t4	change balance to balance+500

Initial value balance: **2000**, final value balance: **2500**

DBA Tasks

Design definition and description of:

- logic diagram
- physical scheme
- sub-schemas and/or views (Data Definition Language)

Maintenance:

- changes for new requirements or efficiency reasons
- (routines: load, copy and restore,
- reorganization, statistics, analysis)

Course content

To recap

The course will cover the following main topics:

- **relational Algebra:** Procedural Query Language
- **database design:** how to guarantee/verify the Third Normal Form (3NF), how to decompose a schema preserving dependencies and information
- **physical organization** of data
- **concurrency** control