Syllabus

Dr. Ahmed E. Khaled

Department of Computer Science

THE UNIVERSITY OF CHICAGO

MPCS 53001- Databases

The Major topics in this course:

- **Introduction**: Essential definitions, Database properties, and Data models
- **Entity-relationship** (ER) model
- **Relational Model and Schema Refinement**
- **Relational Algebra**: Operators to build queries as well as Query processing and optimization
- **Structured Query Language (SQL)**:
  - Essential commands, constraints, and operators
  - Join operator, aggregate functions, and nested queries.
  - Interfacing Java/Python programming languages with SQL
- Categories of data and properties of big data
- **SQL/NoSQL**: Other types of databases
  - **MongoDB**: an example of document-based NoSQL databases
  - **Neo4j**: an example of graph-based NoSQL databases
  - Further topics (as the time allows)

- Make sure you continuously check the course's page on ***Canvas*** for announcements, lectures, and assignments.

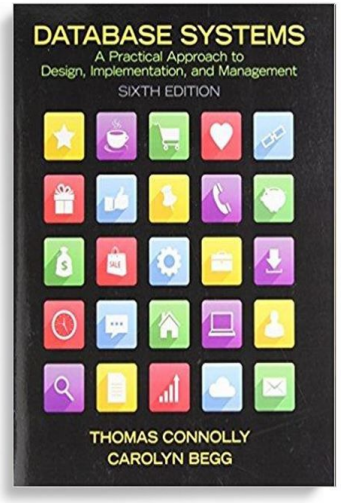
- Your ***active and continuous participation in class is important*** **-** there are ***extra bonus points (up to 5 points)*** for:
    - Asking questions
    - Answering questions
    - Engaging in discussions

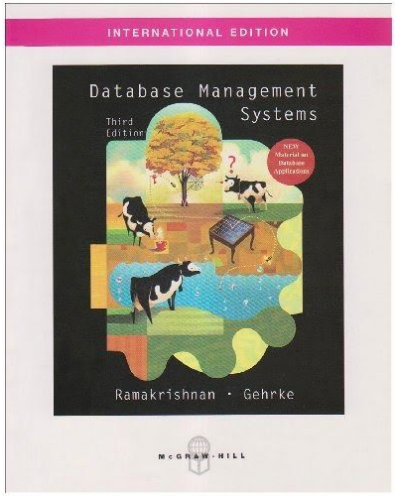
- Your ***grades*** throughout this course are divided as follows:
    - Individual Assignments [homeworks] (65%)
    - Midterm (10%)
    - Group-based Final Project [set of sequential steps] (25%)
    - Class participation [bonus points] (5%)

Course Material

- **Reading material/tutorials will be assigned during the semester**.

- **Optional Textbook:**
  - *Database Systems: A Practical Approach to Design, Implementation, and Management* by Thomas Connelly and Carolyn Begg, 6th Edition.
  - *Database management Systems* by Raghu Ramakrishnan and Johannes Gehrke. 3rd Edition.
  - *MongoDB: the definitive guide: powerful and scalable data storage*. by Shannon Bradshaw, Eoin Brazil, and Kristina Chodorow. 3rd Edition.
  - *Graph databases: new opportunities for connected data*. by Ian Robinson, Jim Webber, and Emil Eifrem. 2nd Edition.





Your total/final grade =

∑ *your grade in the assignments*	*(out of 65 points)* +
∑ *your grade in the Midterm*	*(out of 10 points)* +
∑ *your grade in the Project*	*(out of 25 points)* +
∑ *your grade for bonus points*	*(out of 5 points)*

= ***105/100***!

Check the course's page on Canvas for information about the weekly office hours of the instructor and the TAs, If you have any questions feel free to contact me

Email: ahmedkhaled@uchicago.edu

Department of Computer Science

The University of Chicago

Unit 1: Introduction

Dr. Ahmed E. Khaled

Department of Computer Science

THE UNIVERSITY OF CHICAGO

MPCS 53001- Databases

Topics to Discuss

- Data, Database, and DBMS
- The differences between DBMS and File System
- Data model and Semantic Data model
- The different types/roles for database users
- The main steps to design a database
- The different layers of DBMS abstractions

Data, Database, and DBMS

Data, Database, and DBMS

- Data is everywhere, and almost everything around us accepts and generates data — from our smartphones and social media applications to our home devices, transportation systems, navigation systems, etc.

- Data is just a ***collection*** of unprocessed facts, words, images, and numbers that generate ***meaningful Information*** when understood, organized, interpreted, and put into context.

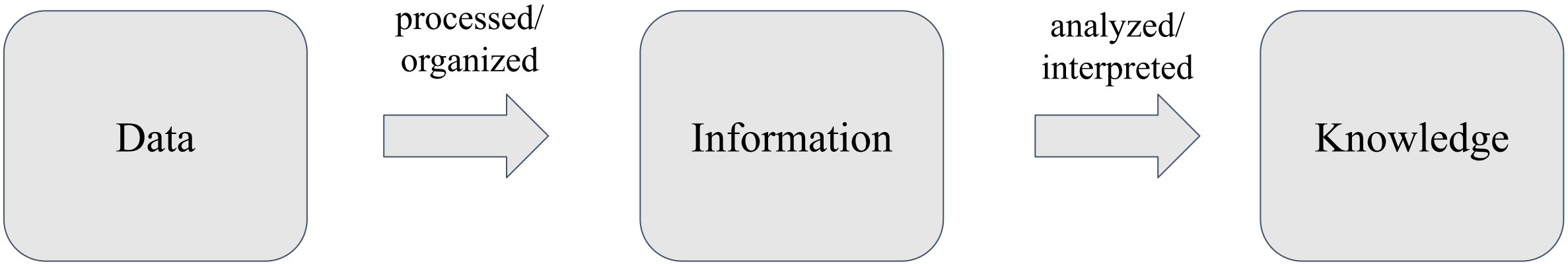
- Such ***Information*** helps us understand the patterns, the trends, as well as the relationships between the different data entities.

- Information, as a critical component of most systems and applications, is used in the ***decision-making process, in the prediction process, and in answering questions***.

Data, Database, and DBMS

"***Information is the oil of the 21st century, and analytics is the combustion engine.***"

— Peter Sondergaard, Global Head of Research at Gartner, Inc.



Data, Database, and DBMS

- When you work with data as a data expert (e.g., data scientist, data analyst, data engineer), system designer, or application developer, it is essential to know the ***fundamental differences between the various data storage and management options***.

- There are different metrics that guide such selection, such as:

  - the type of the data to be stored
  - the size of such data
  - the cost of storing such data
  - the expected operations on the data

Data, Database, and DBMS

- There is a wide range of solutions and options when it comes to storing, managing, and accessing data, including *Databases, Data warehouses, Data Marts, and Data Lakes*.

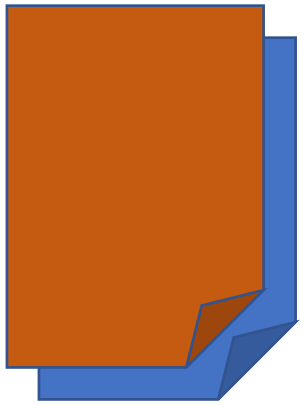
- In this class, we will have a discussion on a set of essential concepts and keywords related to the first point: Databases.

- The scope of the other concepts will be introduced in last chapter.

Data, Database, and DBMS

- A dataset (or data set) is ***a collection of related data items/points*** that can be a series of observations, measurements, or events.

- You can think of a dataset as a snapshot of related data items/points captured at a certain point in time.



Dataset(s)

- A dataset is typically organized in a particular format and the commonly used formats are rows and columns (e.g., CSV), XML, or JSON.

Data, Database, and DBMS

- A dataset is a collection of data items/points, however:
  - It is difficult to ***capture and utilize*** relationships/links between ***different*** datasets

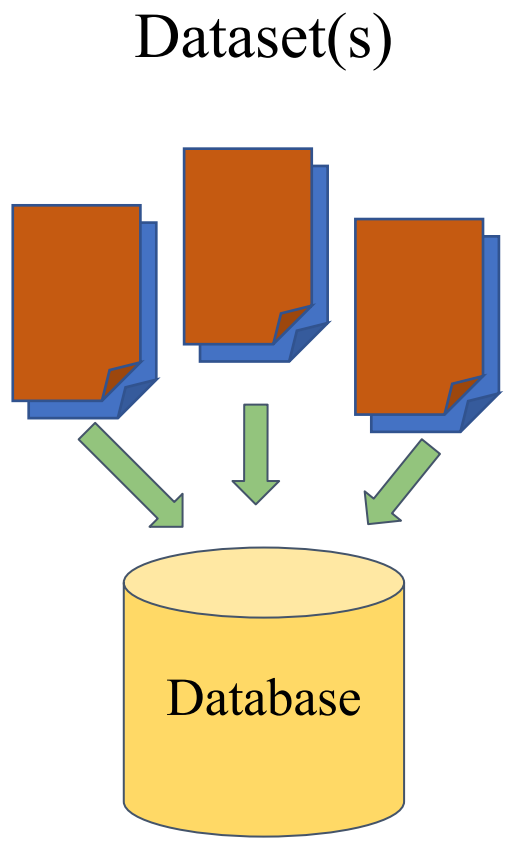
1	22	5	13
2	21	6	7
3	19	6	6
4	15	7	6
5	18	6	7
6	29	5	8
7	34	4	9
8	30	8	8
9	19	9	7
10	7	12	15

Dataset(s)

- A dataset is usually used as ***an input*** to statistical tools for data analysis-related tasks and predictive models.

Data, Database, and DBMS

- A database is an organized collection or container of data, typically from ***multiple related datasets***.



- Most of today's websites, applications, and platforms utilize databases - for example:
  - Online Ecommerce (e.g., Amazon) - to store information about the consumers, products, preferences, and sales.
  - Online Bookstore (e.g., Barnes & Noble) - to store information about the books, related-products, and available branches.
  - ***Do you have more example ?***

Data, Database, and DBMS

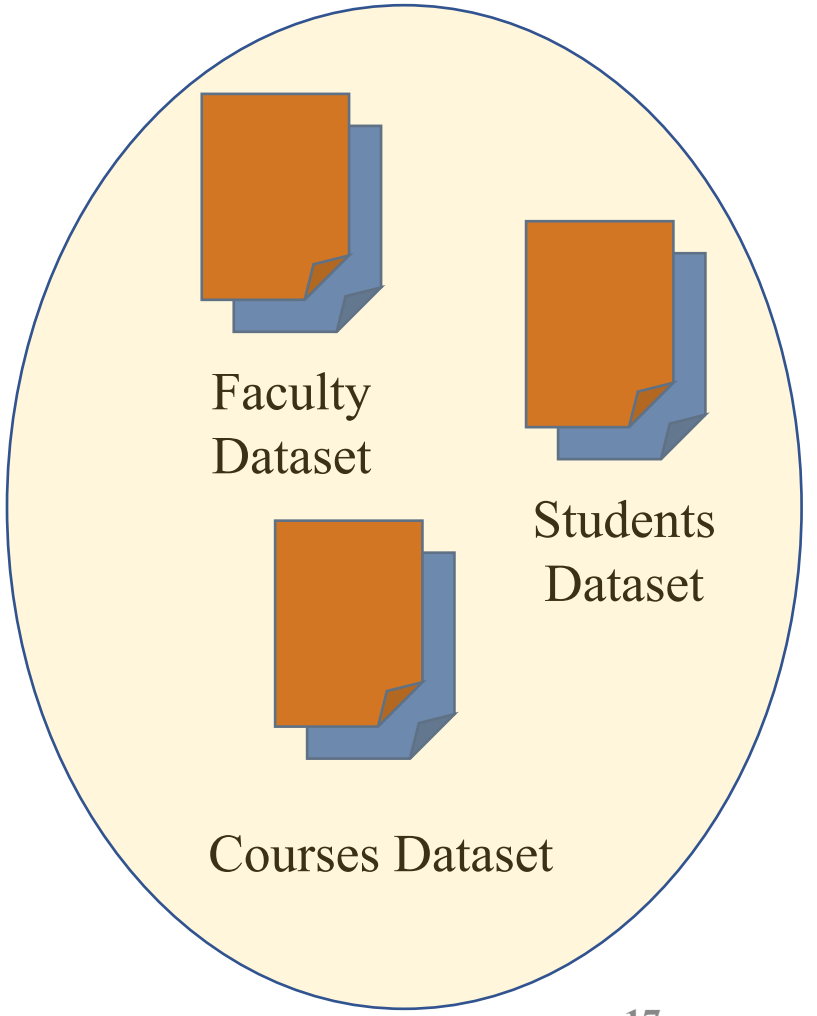
A ***university database*** (as an example) might contain information about:

Entities: students, faculty, courses, and classrooms

Attributes of entities: student name, course number, faculty web page, and classroom location

Relationships between entities: students' enrollment in a course, faculty teaching courses, and the use of classrooms for the courses.

University Database
"Collection of university-related data"



Data, Database, and DBMS

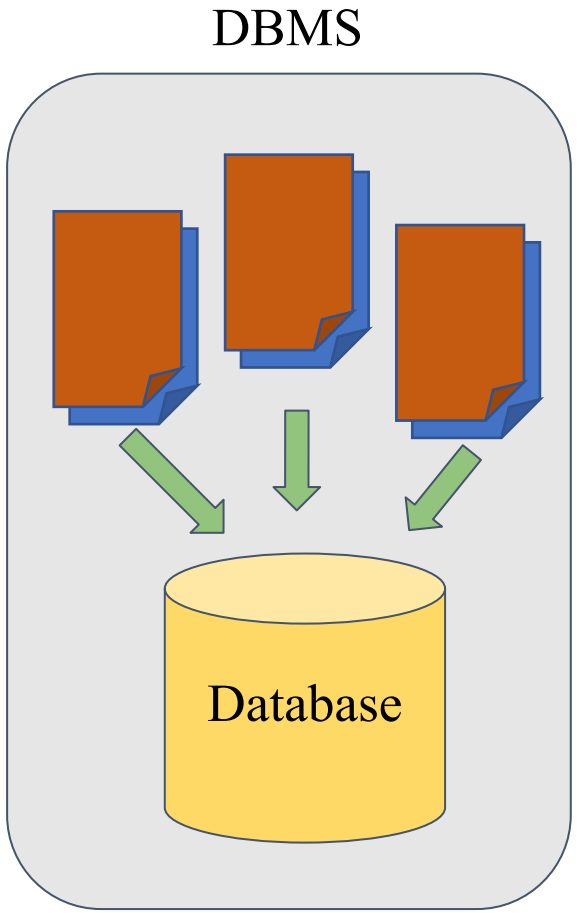
- The value of such organized collection of data is ***widely recognized as an asset***.

- To get the most out of our collection of data, we need ***tools*** to manage data and extract useful information in ***an efficient and timely fashion way***.

- Otherwise, the cost of acquiring and managing data ***far exceeds*** the value derived from it.

Data, Database, and DBMS

- The basic and core features supported by any database are the Create, Read, Update, and Delete (CRUD) operations.
- The CRUD operations are usually performed by users via a ***query language*** through a Database Management System (DBMS).
- A DBMS is a ***software tool designed to help*** users, admins, and developers to build databases and to maintain a large collection of data.
- We use DBMS to create databases to ***store*** data/dataset and then ***ask*** questions (queries) about the stored data.

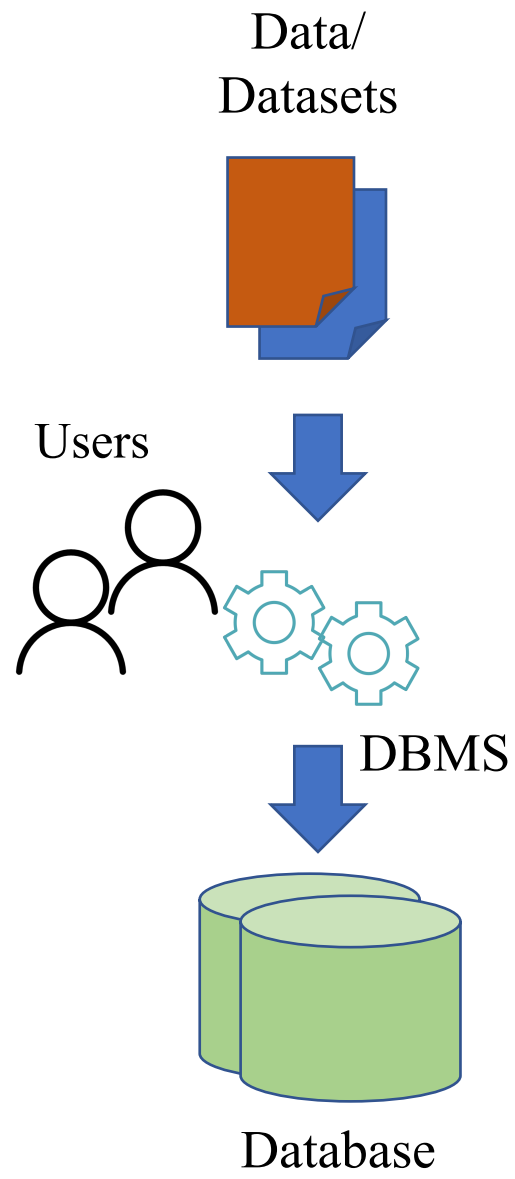


Data, Database, and DBMS

- A DBMS offer a wide set of features that include: efficient and optimized data storage, accessibility, retrieval of data, data protection and security, and multi-user access.

- As we will discuss in this course, different types of data and requirements lead to ***different types of databases*** (e.g., relational, graph-based, document-based), and for each type there are ***different DBMSs and query languages***.

- MySQL, Oracle, Cassandra, Neo4j, InfluxDB, and MongoDB are examples of DBMS.



First solution: File System

An alternative to a DBMS is to use the operating system's file system (files and folders) to organize your collection of data, however we will face a set of challenges:

- Data ***redundancy*** (there can be repetition of same data in different files).
- We may face multiple file ***formats*** (different files with different format) that may increase the complexity of accessing data.
- We must write query-dependent programs to answer ***each question (query) a user may want to ask***. To answer a query, your program must be designed and developed to access specific set of files and link certain data items/points within these files.

First solution: File System

An alternative to a DBMS is to use the operating system's file system (files and folders) to organize your collection of data, however we will face a set of challenges:

- There are limited options to ***protect data*** from unauthorized access. If you open a file that contains information about employees, it might be difficult to hide certain attributes (e.g., addresses, salaries).

- We must ensure that the data is restored to a consistent state ***if the system crashes***. Imagine you are working on a file (adding, deleting, updating entries) and the system crashes (e.g., due to a power problem) – you need to know which of these operations were completed successfully and which of these operations need to be un-done.

- Imagine two users are editing the same data record (e.g., of a certain item) at the same time; we must protect the data from ***inconsistent changes*** made by different users accessing the data concurrently.

Second solution: DBMS

By storing data in a DBMS, we can utilize a set of features to manage data in a ***robust and an efficient manner***.

- ***Data Integrity:*** DBMS can enforce certain constraints on the data (e.g., the new employee's salary does not exceed certain value, an age can not be a negative value).

- ***Security:*** DBMS enables access control that governs data visibility and access to different classes of users (e.g., only admins can view/edit salaries).

- ***Concurrent Access and Crash Recovery:*** DBMS schedules concurrent accesses to the data and protects data from system failures (crashes).

Second solution: DBMS

By storing data in a DBMS, we can utilize a set of features to manage data in a ***robust and an efficient manner***.

- ***Query Language:*** using DBMS, we can answer questions about data using query languages (e.g., SQL, Cypher), and no need to design and develop an application per query.

- ***Efficient Data Access:*** DBMS utilizes a variety of techniques (e.g., indexing) to store and retrieve data efficiently.

- ***Data Independence:*** Applications focus only on the required data, where DBMS provides an abstract view of the data that hides details on how such data is represented and stored.

Is there a reason ***not to use*** a DBMS? - Yes!

- DBMS is a complex piece of software that is optimized for certain kinds of workloads (e.g., answering complex queries, handling concurrent requests from different users).

- Such complex software may not be adequate for ***certain applications*** with a few well-defined operations where ***an efficient custom code*** can be used instead

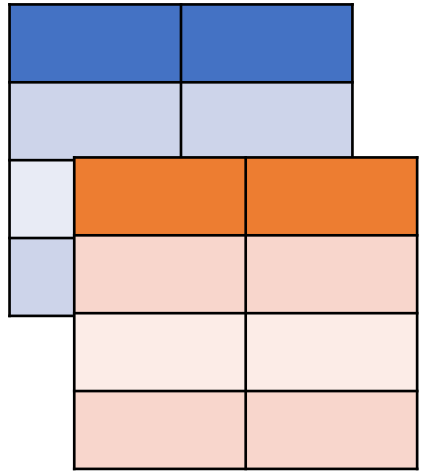
- For example, If you have data about products in a file and you are just interested about answering specific question (e.g., average price) - you may not need such complex DBMS, while building custom code to answer this question is more efficient.

Data Model vs Semantic Data Model

Data model

What is a data model?

- Collection of high-level data description constructs (e.g., tables, trees, graph) that hide many ***low-level details about the physical storage of data***.
- A DBMS –through data model- allows the users to focus on organizing and structuring the data rather than on how such data is ***physically stored***.
- One of the commonly used data models is the ***relational data model*** – where data is stored in tables and the different tables are linked together.

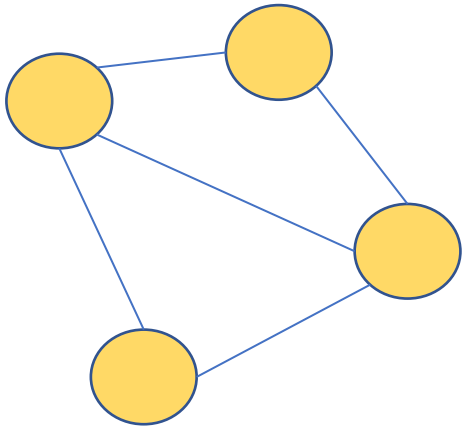


Data-model
(e.g., Relational model)

Semantic Data model

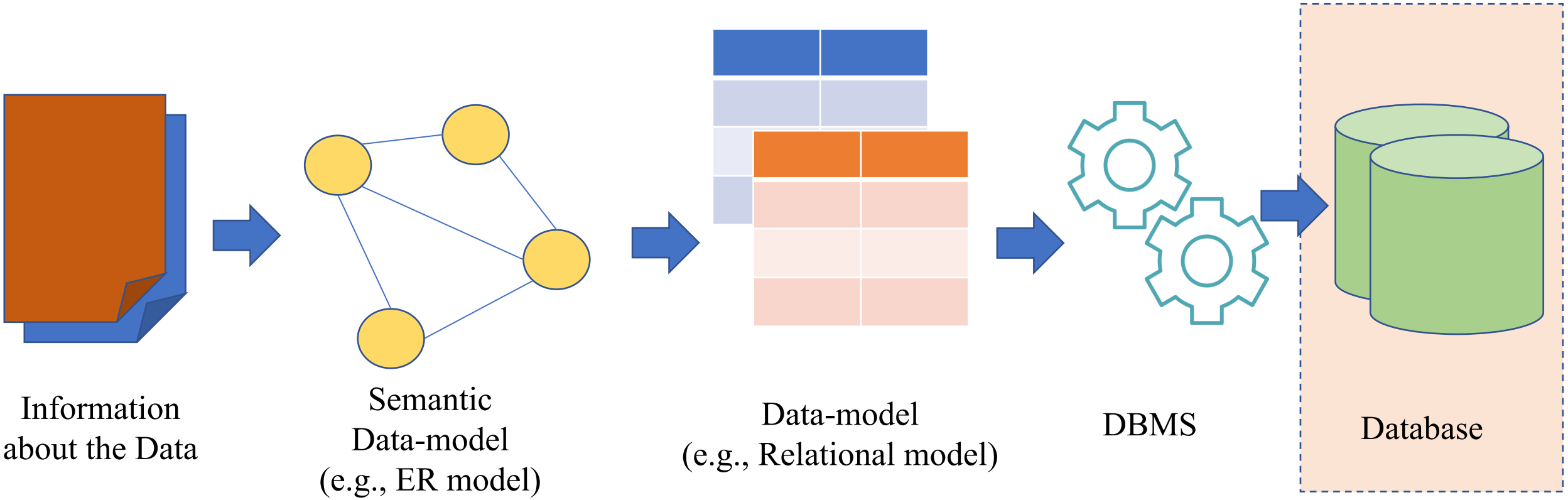
What is a semantic data model?

- More abstract model (compared to the data model) for a ***good initial description*** of the data.
- Defining such model is the start point in ***designing a database***, which is then ***translated into a data model*** that a DBMS implements.
- A widely used semantic data model is the ***entity-relationship (ER) model*** that allows us to define and visualize the different entities and the relationships among them.



Semantic Data-model
(e.g., ER model)

Semantic data model vs data model



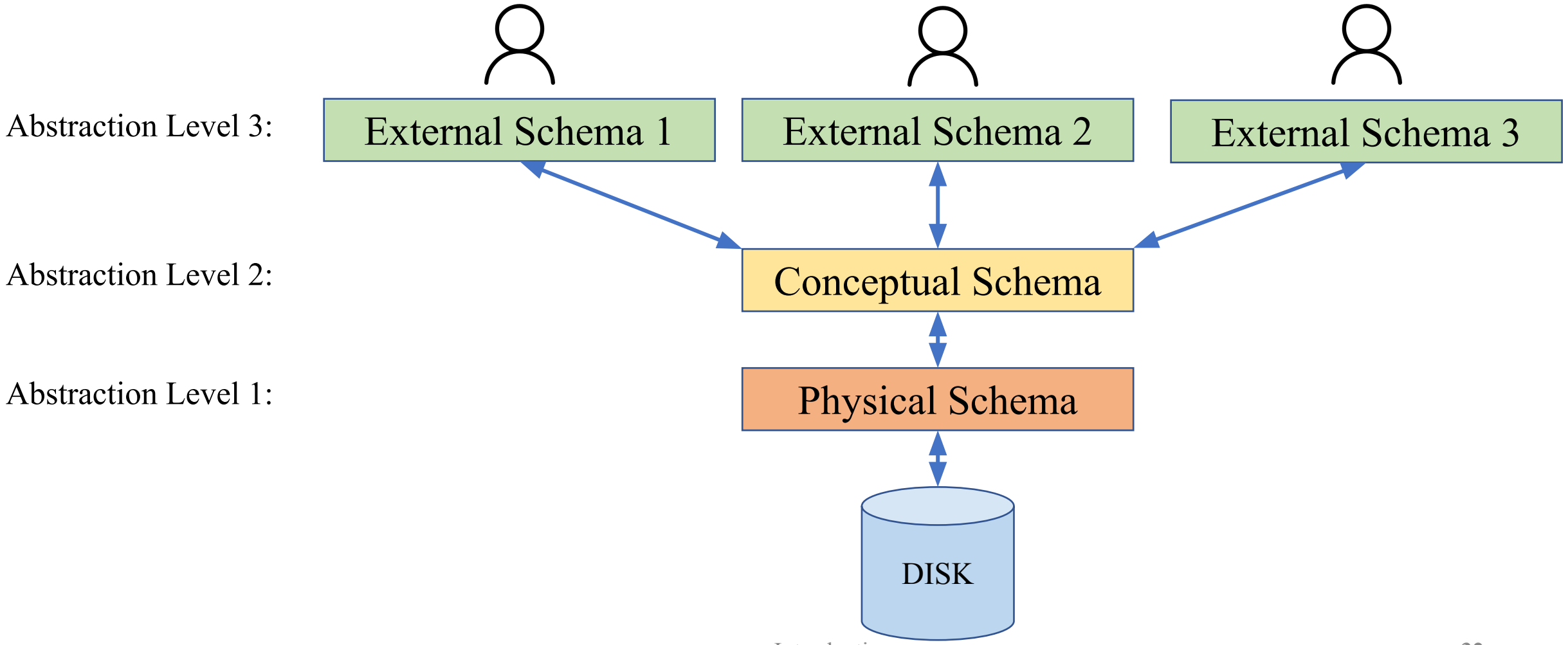
Important keywords so far

- Database
- DBMS
- File systems
- Data model (Relational data model)
- Semantic data model (ER diagram)

Levels of Abstraction in a DBMS

Levels of Abstraction in a DBMS

The data in a DBMS is described on ***three levels of abstractions***, each level offers certain functionalities and services and also hides certain implementation details.



Levels of Abstraction in a DBMS

- The ***Physical schema*** describes how the data items/points in the conceptual schema are physically stored on secondary storage devices (e.g., hard disk) and in which format.

- This schema utilizes different techniques (e.g., indexes) to organize data storage for ***efficient retrieval of information***.

Levels of Abstraction in a DBMS

- The ***conceptual*** (or logical) ***schema*** describes the data in terms of a data structure or a data model that can be utilized to store and access data items/points (e.g., relational data model, document model, graph model).

- According to the type of the data to be stored and the application we target, we have ***databases*** that support:
  - ***Static schemas***: a predefined and rigid structures to host data
  - ***Dynamic schemas***: more flexible structures to host data

- Every schema type or data model has properties, advantages, and disadvantages; as we will discuss throughout this course.

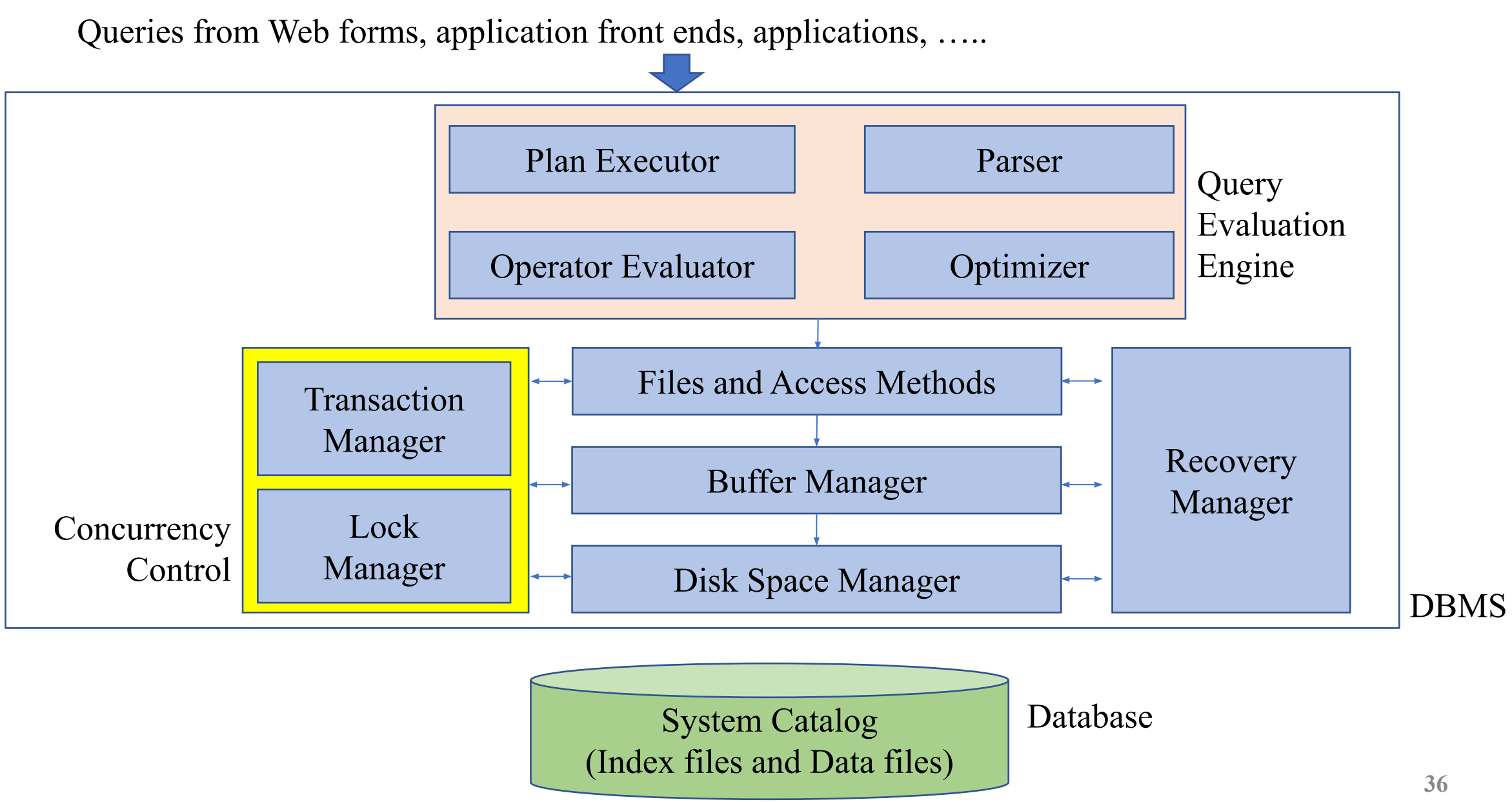
Levels of Abstraction in a DBMS

- The ***External schema*** allows the data accessibility to be ***customized (and authorized)*** at the level of individual users or groups of users.

- Each external schema consists of ***view(s)*** - a way to ***access*** data with some constraints (e.g., restrict employees to only view the names of company's staff, restricts that the salary of employees can only be accessed by the HR).

- Any database may have ***several external schemas*** - each is tailored to a group of users.

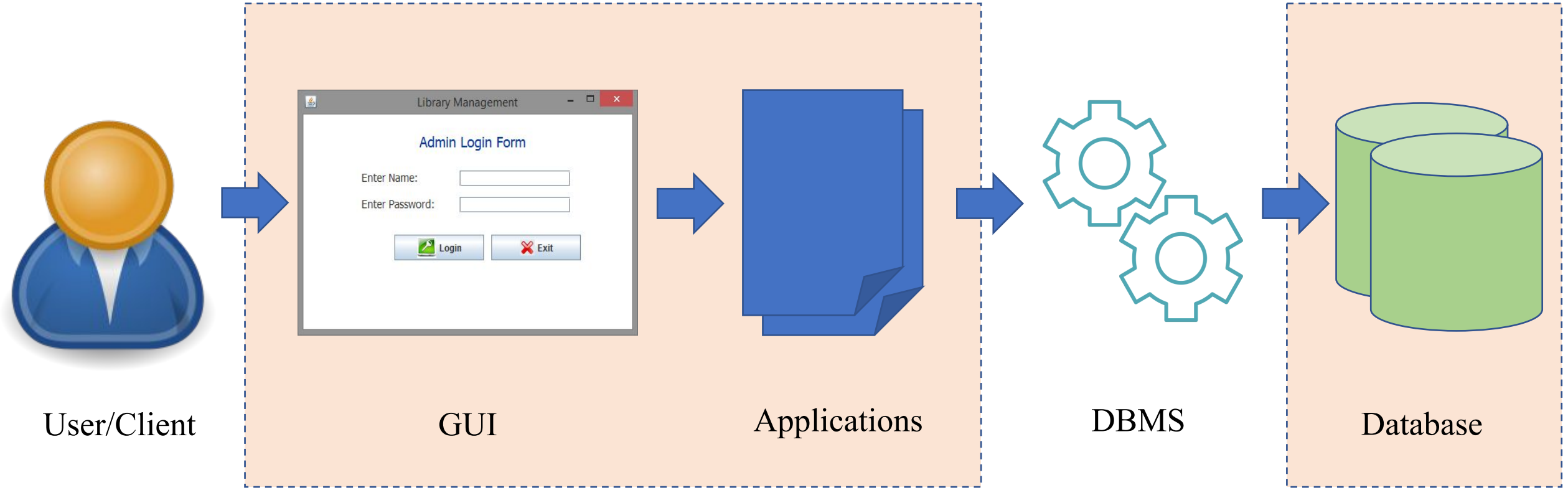
A general architecture of a DBMS



Who are the Database Users?

Who are the Database Users?

Imagine an online ecommerce system with a backend and frontend:



Who are the users of the database utilized in this system?

Who are the Database Users?

- Database ***Administrator***
  - ✔ authorizes an access to the database, coordinates and monitors the use
  - ✔ acquires software and hardware resources as needed

- Database ***Designers***
  - ✔ identify the data to be stored in the database
  - ✔ choose the appropriate structures (data model) to represent and store the data
  - ✔ develop external schemas on the database that meet the users' requirements

In this class, we are the ***database designers*** who will design and build databases

Who are the Database Users?

- ***Application*** Programmers
  - ✔ Determine the requirements of end-users and implement applications/interfaces that connect and interact with DBMS.
  - ✔ access the database –through the developed applications/interfaces- to query, update, and generate reports

In this class, we are also the ***application programmers*** who will develop programs/interfaces that interact with a DBMS

Who are the Database Users?

- Naïve ***Users***
  - ✔ Invoke the developed applications (programs) to perform certain tasks on the database.

- ***Database systems*** Programmers
  - ✔ Design and implement the database systems themselves.
  - ✔ Design and implement DBMS modules (for implementing data access, concurrency control, recovery, and security) features and functionalities of DBMS.

How can we design a database ?

How can we design a database ?

The process of designing a database can be divided into ***six main sequential steps***:

Step 1- ***Requirements Analysis***

- Understand the target application or scenario that requires a special focus on what ***data*** to be stored and what ***operations*** are most frequent and subject to performance requirements.

- Usually an ***informal process*** that involves a deep understand of what the users want from the database and involves ***discussions with user groups***.

How can we design a database ?

Step 2- ***Conceptual Database Design***

- The Information gathered in step-1 is used to develop a ***high-level description*** of the data to be stored, the relationships between the different data items/points, as well as the ***constraints*** known to hold over this data.

- This step is often carried out using the a high-level (or semantic) data model like the ***Entity-Relationship (ER) model*** that is used to provide an initial design.

Designing an ER diagram is our focus in Unit 2

How can we design a database ?

Step 3- ***Logical Database Design***

- Converting the ***conceptual database design*** into a lower-level design for the database using ***the data model*** of the chosen DBMS.

- In this step we convert the designed high-level diagram (of Step 2) into a ***data model (e.g., relational model)***

Designing a relational data model, using relational algebra, and using query languages for queries is our focus in Unit 3, 4, and 5 respectively

In this course, we will also discuss and use other types of data models and DBMSs in Units 6, 7, and 8

How can we design a database ?

Step 4- ***Schema Refinement***

- Analyzing the collection of relations in our relational database schema (from step 3) to identity ***potential problems*** and refine it through a set of ***theories***.

- In this step we will learn about the theory of ***normalizing relations*** – restructuring our database to ensure some desirable properties.

We will discuss a set of normalization techniques in Unit 3

How can we design a database ?

Step 5- ***Physical Database Design***

- Considering ***typical workloads*** that our database must support and further refine the design to ensure that the new design meets ***desired performance criteria*** (e.g., access time, memory cost).

- This step may involve building ***indexes*** and ***clustering*** some data or it may involve a ***substantial redesign*** of part of the database schema

How can we design a database ?

Step 6- ***Application Design***

- Any software project that involves a DBMS must consider aspects ***of the application that go beyond the database itself*** (e.g., the design and usability of the user interface).

- In the implementation phase, we will use programming languages (e.g., Java, python, C) to design and implement applications/interfaces for the users to access data from the ***DBMS.***

Important keywords so far

- Data Model vs Semantic Data Model
- The levels of abstractions in a DBMS
- The different users of a database/database system
- The main steps to design a database