# View of Data

Dr. Saad Adnan Abed

Outline

- View of data
- Data models categories
- Database Languages

# View of data

A database system:

Allow users to access and modify these data.

Provides users with an abstract view of the data.

The system hides certain details of how the data are stored and maintained.

This is the main purpose of Database systems

# Data abstraction

Usable database system must retrieve data efficiently.

To achieve this, designers use complex data structures to represent data in the database

Usually database users are not computer trained.

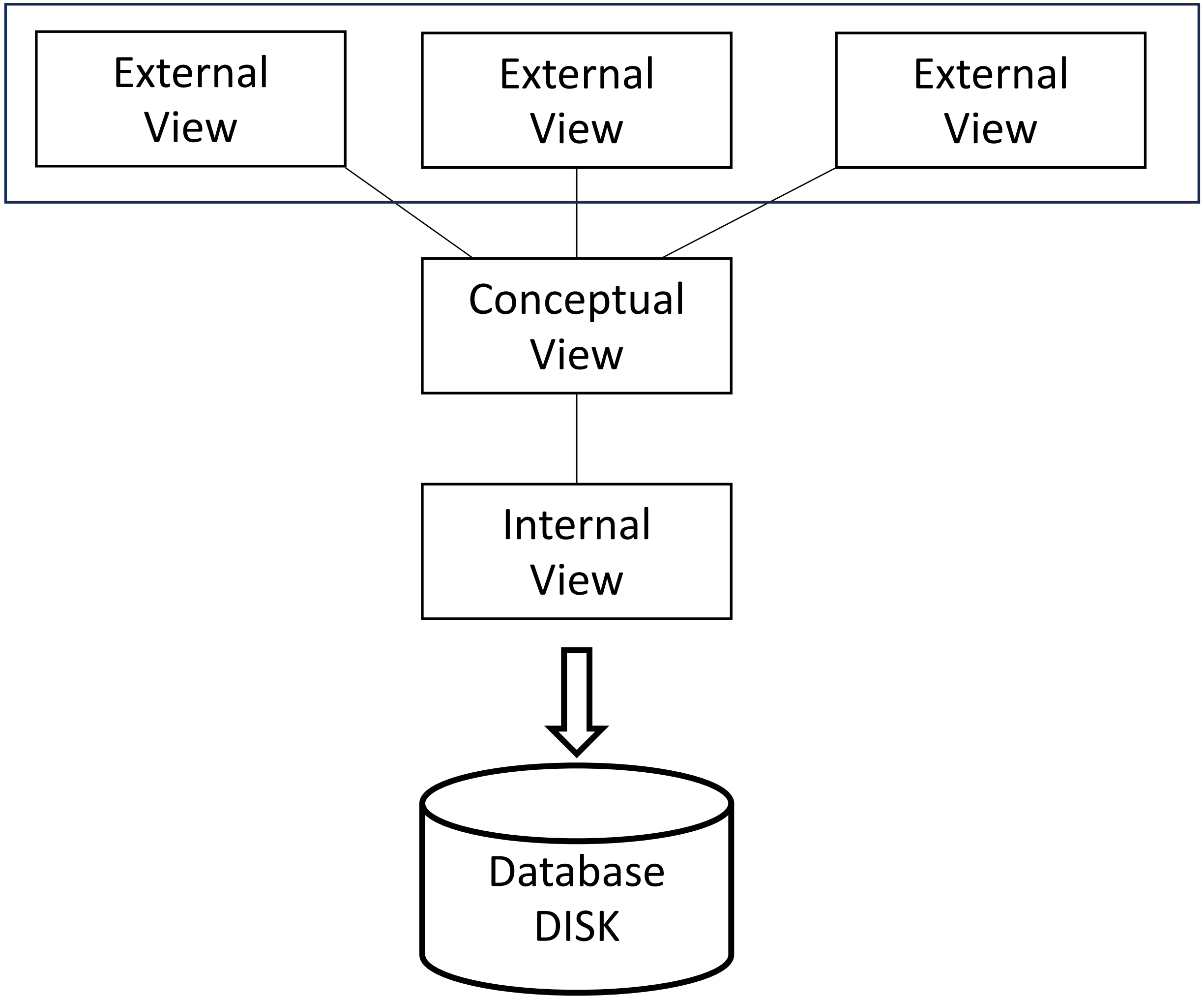
Thus, developers hide the complexity from users through several levels of abstraction.

# Levels of abstraction in a DBMS

**External Schema**

**Conceptual Schema**

**Internal Schema**



**Several views of the database**

# Levels of abstraction in a DBMS

**Physical level (or Internal View / Schema):** The lowest level of abstraction describes how the data are actually stored. The physical level describes complex low-level data structures in detail.

**Logical level (or Conceptual View / Schema):** describes what data are stored in the database, and what relationships exist among those data. The logical level thus describes the entire database in terms of a small number of relatively simple structures.

**View level (or External View / Schema):** describes only part of the entire database.

# Levels of abstraction in a DBMS

Many high-level programming languages support the notion of a structured type. For example, we may describe a record as follows:

```
struct Instructor {
    string ID;
    string name;
    string dept_name;
    double salary;
};
```

A university organization may have several such record types, including

- ***department***, with fields *dept_name*, *building*, and *budget*
- ***course***, with fields *course_id*, *title*, *dept_name*, and *credits*
- ***student***, with fields *ID*, *name*, *dept_name*, and *tot_cred*

# Levels of abstraction in a DBMS

**At the physical level**, an instructor, department, or student record can be described as a block of consecutive storage locations. The compiler hides this level of detail from programmers.

**At the logical level**, each such record is described by a type definition, as in the previous code segment, and the interrelationship of these record types is defined as well.

**At the view level**, computer users see a set of application programs that hide details of the data types.

# Instances and Schemas

Databases change over time as information is inserted and deleted.

Each variable has a particular value at a given instant.

**Instance:** is the collection of information stored in the database at a particular moment.

A database schema corresponds to the variable declarations (along with associated type definitions)

**Schema**: is the overall design of the database. Schemas are changed infrequently, if at all.

# Data models

Underlying the structure of a database is the **data model**: a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints

A data model provides a way to describe the design of a database at the physical, logical, and view levels.

# Data models categories:

**Relational Model**. The relational model uses a collection of tables to represent both data and the relationships among those data.

**Entity-Relationship Model**. The entity-relationship (E-R) data model uses a collection of basic objects, called *entities*, and *relationships* among these objects.

**Object-Based Data Model**. an object-oriented data model that can be seen as extending the E-R model with notions of encapsulation, methods (functions), and object identity.

# Data models categories:

***Semi-structured Data Model.*** The semi-structured data model permits the specification of data where individual data items of the same type may have different sets of attributes.

The **network data model** and the **hierarchical data model** preceded the relational data model. These models were tied closely to the underlying implementation, and complicated the task of modeling data.

# Database Languages

A database system provides a **data-definition language** to specify the database schema and a **data-manipulation language** to express database queries and updates.

In practice, the data definition and data-manipulation languages are not two separate languages; instead they simply form parts of a single database language, such as the widely used SQL language.

# Data-Manipulation Language

**Data-manipulation language (DML)** is a language that enables users to access or manipulate data as organized by the appropriate data model.

The types of access are:

- Retrieval of information stored in the database
- Insertion of new information into the database
- Deletion of information from the database
- Modification of information stored in the database

# Data-Definition Language (DDL)

We specify a database schema by a set of definitions expressed by a special language called a data-definition language (DDL). The DDL is also used to specify additional properties of the data.

We specify the storage structure and access methods used by the database system by a set of statements in a special type of DDL called a data storage and definition language.

These statements define the implementation details of the database schemas, which are usually hidden from the users. The data values stored in the database must satisfy certain consistency constraints.

# Discussion