Chapter 1
Database Approach and concepts



Some questions

- What are before database? problems and solutions?
- What is in your opinion the best way to organize the data and where?
- How will we search the data to find what we need?

1. Before Databases

Before databases, information was stored in **paper files** or **file-based systems (FBS such as .txt, .csv, .xml, binary)**.

• **Problems:** ◦ Program data dependance. ◦ Redundancy (same data stored in many files). ◦ Inconsistency (updates not reflected everywhere). ◦ Concurrency issues. ◦ Poor search (manual or limited access). ◦ Low security. ◦ Separation and isolation. ◦ Data anomalies and modifications problems: update – deletion-….	• **Solutions:** ◦ Centralized **databases**. ◦ **DBMS** to manage access and consistency

2. Database World (1)

a) **Data**: Raw facts that can be recorded and that have implicit meaning

(*"John, 25, Paris"*).

Types of data:

- **Structured:** numbers, text in tables.
- **Semi-structured:** XML, JSON.
- **Unstructured:** images, audio, video.

b) **Information**: Data itself has no meaning without metadata which describes data. Processed data (*"Average age = 25"*).

2. Database World (2)

Data vs information

- Data is what you **store** in database. Information is what you **retrieve** from database.

c) **Metadata (system catalog/ data dictionary)**: Data about data (table name, column type, size).

data:

Name	City	Birth
Khaled	Beirut	01/01/1996
Charbel	Beirut	20/05/1997

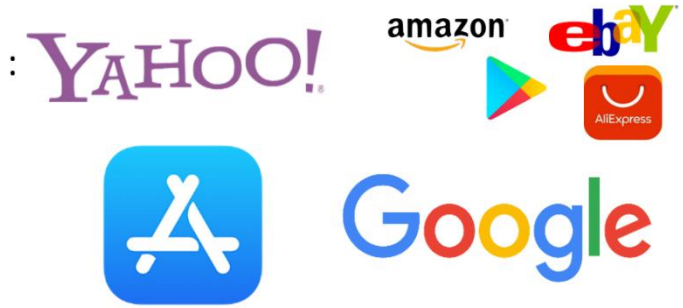
Metadata:

- name-string, length<10
- city-string,length<15
- birth-Date,format DD/MM/YY

2. Database World (3)

d) **Database** is a collection of information that exists over a long period of time, often many years.

- **Database is a large and organized collection of logically related data and a description of this data (metadata) designed to meet the information needed of an organization. A database may be of any size and complexity.**
- Databases are used to maintain and manage Data found in:
  - Websites
  - Corporations
  - Research institutes
  - Companies



Examples: Enroll in a course; Transfer money ; Shop on Amazon; …

- **Technical definition of Databases: A database is a collection of data managed by a DBMS (Database Management System**

2. Database World (4) - Database contents

- **Entity (table or relation)**: a distinct object in the organization (person-place-thing-event). Ex: student, courses.
- **Attribute (fields or features or column)**: Characteristic that describes some aspect of the entity that we wish to record. Example: student_id, course_credit.
- **Rows (instance or tuple)**
- **Relationship**: an association between entities. **Example: Sami is studying database.**
- **Actual data** (tables, rows, columns).
- **Metadata** (schemas, constraints).
- **Application programs**.

Example of database

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course _number	Credit_hours	depatement

SECTION

Section_id	Course_nb	Semester	Year	Instruction

GRADE-REPORT

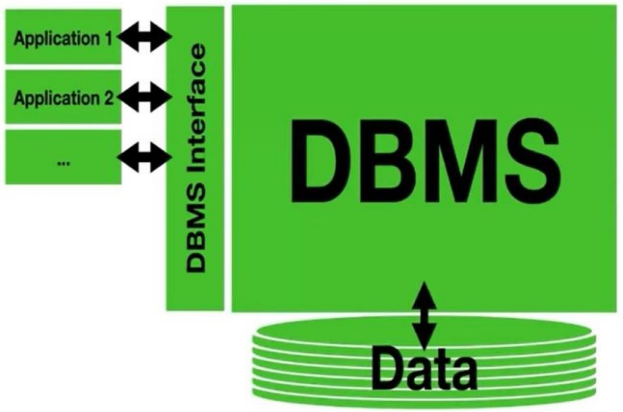
Student_nb	Section_id	Grade

PREREQUISTE

Course_nb	Prerequiste_nb

3. Data Base Management System (DBMS)

3.1. DBMS-Definition



- The power of databases comes from powerful software that has been developed over several decades and is called a **Database Management System** or **DBMS.**
- DBMS is a **software system** that enables users to create, maintain and query the database.

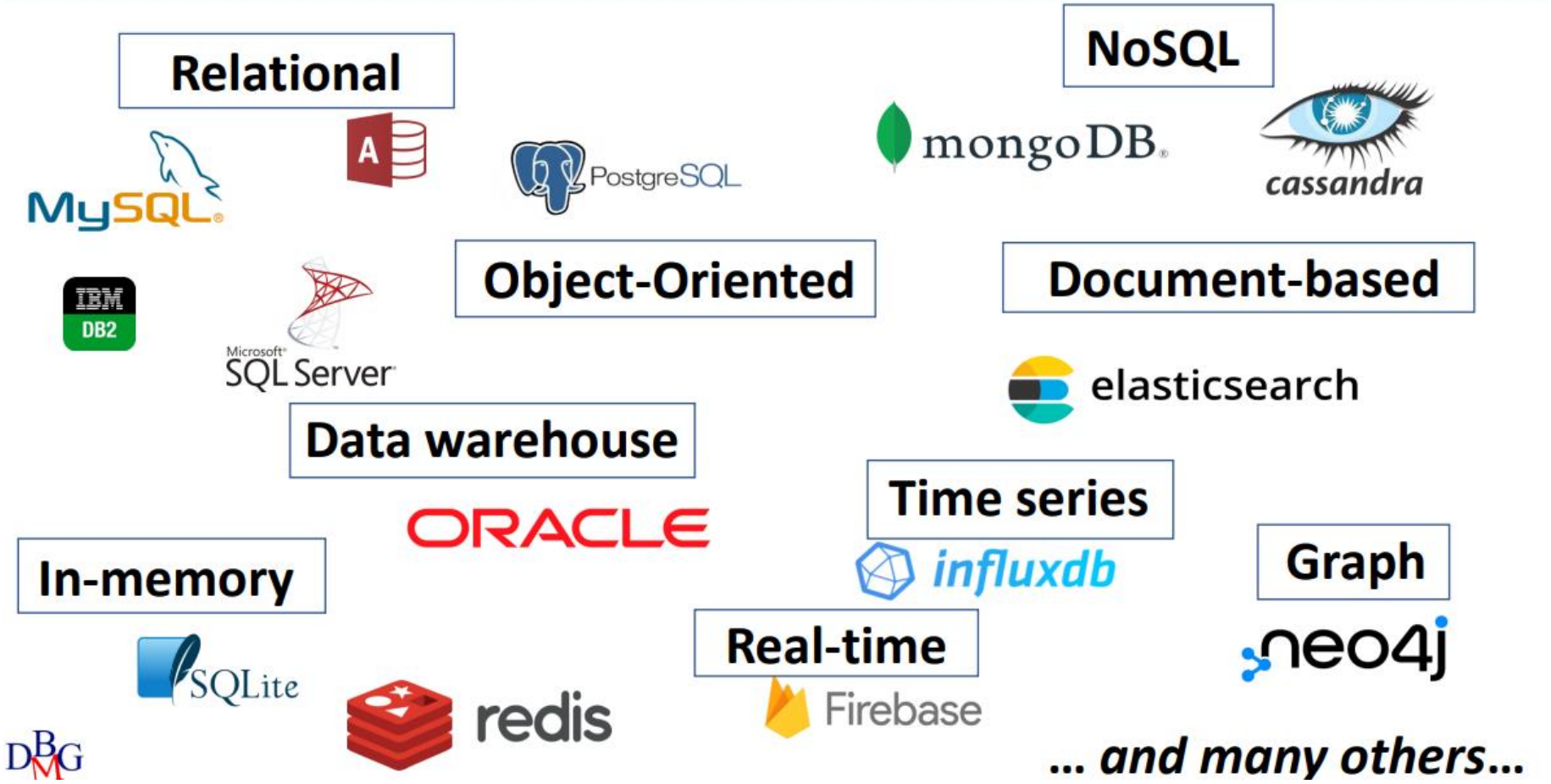
DBMS is used for creating and managing large amounts of data efficiently and allowing it to persist over long periods of time, safely.

DBMS software to create, manage, and interact with databases.

DBMS examples: Microsoft access – Microsoft SQL Server – Oracle- MySQL -…

DBMS Functions: define data, manipulate data (CRUD), control access, ensure integrity.

3.2. DBMS types



3.3. Evolution of DBMS

1.Early Database Systems

- Evolved from file systems and date back to 1960's
- Not all DBMS properties are offered
- First important applications of DBMS:
  - Banking systems
  - Airline Reservation systems
  - Corporate record keeping

They all use graph tree structure standardized in CODASYL (Committee on Data systems and Languages)

2. Relational Database system (MYSQL-PostgreSQL-…)

- Relational Model was born in 1970 by a famous paper written by Ted Codd
- Data organized as **tables** called *relations*
- The programmers are not involved with the storage structure

Queries are expressed in a high-level language called SQL

Relation: *Movies*

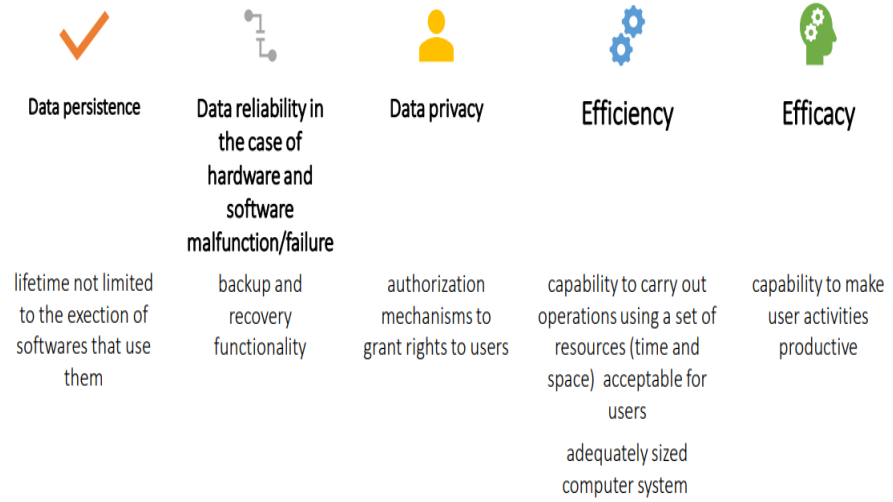
	Title	Year	Genre	Director
Tuple 1 →	Avatar	2009	Adventure	James Cameron
Tuple 2 →	W Halla2 la wein?	2011	Drama/comedy	Nadine Labaki

3.NoSQL Database system (JSON-XML-..)

- NoSQL ("non-SQL" or "not only SQL") databases store data in forms different from relational tables.
- The four main types of NoSQL databases:
  - Document databases
  - Key-value stores
  - Column-oriented databases
  - Graph databases

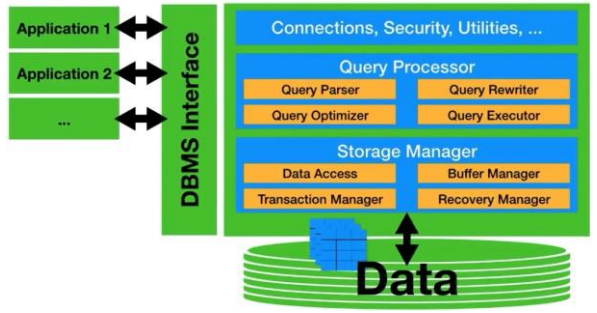
3.4. DBMS properties

1. DBMS allows users to create new databases and specify their structure
2. Allows users to Query the data (retrieve data) and modify it (CRUD = **C**reate – **R**ead- **U**pdate and **D**elete)
3. Supports storage of large amounts of data over a long period of time (data management in secondary memory): Efficiently – Reliably – Securely Conveniently
4. Ensures durability and recovery.
5. Data sharing between applications and user: a database is an integrated resource, shared by several company sectors.
6. Controls access to databases: Assign rights to users
7. Allows execution of transactions: isolation and atomicity
8. Reduction of data redundancy
9. Reduction of data inconsistency



- **Due to advances in technology, we can store gigabytes on single disks now. DBMS can be run on personal computers now**

3.5. DBMS components



Data- Hardware: PC or network of computers - **Software:** DBMS, OS, applications programs

Component	Role / Function	Example
1. Storage Manager	Responsible for managing the physical storage of data on disks. It organizes how data, indexes, and logs are stored and retrieved efficiently.	Reads/writes pages to disk; handles data files.
2. Query Processor	Interprets and executes SQL queries. It parses, optimizes, and runs queries to get results from the database.	Converts `SELECT * FROM Student;` into efficient operations.
3. Transaction Manager	Ensures transactions follow the **ACID** properties — atomicity, consistency, isolation, and durability. Handles concurrency and recovery.	Guarantees safe money transfer between accounts.
4. Metadata Catalog (Data Dictionary)	Stores information **about** the database (data types, table names, constraints, indexes, etc.).	"Student" table has 5 columns, "ID" is primary key.
5. Authorization and Integrity Manager	Manages security and data integrity rules — who can access or modify what data.	Ensures only admins can delete a record.
6. Concurrency Control Manager *(sometimes part of Transaction Manager)*	Ensures multiple users can work simultaneously without interfering with each other.	Two users updating the same table won't cause corruption.
7. Recovery Manager *(also part of Transaction Manager)*	Restores the database to a consistent state after a crash or failure using logs or checkpoints.	If the system crashes mid-update, it rolls back safely.
8. User Interface / API Layer	Allows users and applications to interact with the DBMS (through SQL clients, GUIs, or APIs).	MySQL Workbench, Python `sqlite3` library, JDBC in Java.

Procedures: Instructions and rules that should be applied to the design and use of the DB and DBMS

Summary: The Storage Manager brings the data from disk -The Query Processor reads yo ur requests - The Transaction Manager ensures operations are safe-Recovery Manager fixes issues after crashes - The Catalog remembers what the database looks like."

3.6. DBMS components- Transaction Manager

Transaction: A transaction is a sequence of one or more database operations that form a single logical unit of work.

Examples: Transferring money between accounts-Enrolling a student in a course.

Each transaction must be executed completely or not at all.

Transaction Manager: A component of the DBMS that ensures transactions are executed safely and correctly, even in case of: Power failure - System crash -Concurrent users

It coordinates:

- Concurrency control (so users don't interfere with each other).
- Recovery management (restores DB to a consistent state after failure).

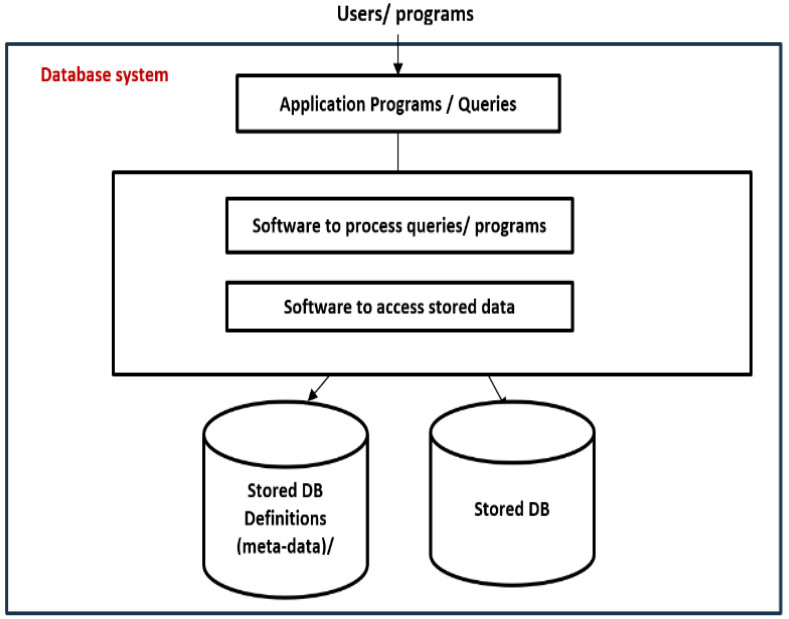
ACID Properties

To guarantee correctness, every transaction must satisfy ACID**:**

Property	Description	Example
A - Atomicity	All operations in a transaction succeed or none do.	Money transfer: if debit fails, credit is undone.
C - Consistency	A transaction brings the database from one valid state to another.	Balance totals remain the same before and after.
I - Isolation	Concurrent transactions do not affect each other's results.	Two users booking seats don't see partial updates.
D - Durability	Once a transaction is committed, it remains even after system failure.	After power loss, saved data persists.

3. Data Base System

Database system = DB + DBMS + application program or users



A DBMS allows users to query and modify data:

- Example of a user: A java application, a web application, etc..
- Query data: Retrieve and read data from the database
- Modify data: Insert new data (or Create), update and delete
- Acronym CRUD: Create, read, update and delete

Database application is a collection of data and programs that interact with the database by issuing an appropriate request (typically an SQL statement) to the DBMS. It's build on top of DBMS to statisfy end users special requirements and preferences. Examples: App of a supermarket- Aireline reservation – Using internet – Credit card purchases

In this course, we will discover **relational databases**

Example of a relational database School

- A relational database is a collection of relations (Tables)

Relation **Teachers**

TeacherID	name	age
1	John smith	33
2	Jane white	40

Relation **Courses**

courseCode	CourseName
CSC375	Database Management systems
CSC380	Microcontrollers

Relation **Teach**

courseCode	TeacherID
CSC375	1
CSC375	2
CSC380	1

In the above example, the database school has 3 tables. in Table teachers, we have 2 tuples denoting 2 teachers, john and jane. In table courses, we have 2 courses, and in table teach, we have 3 tuples that tell us that john teaches CSC375 and CSC380, while jane only teaches 375

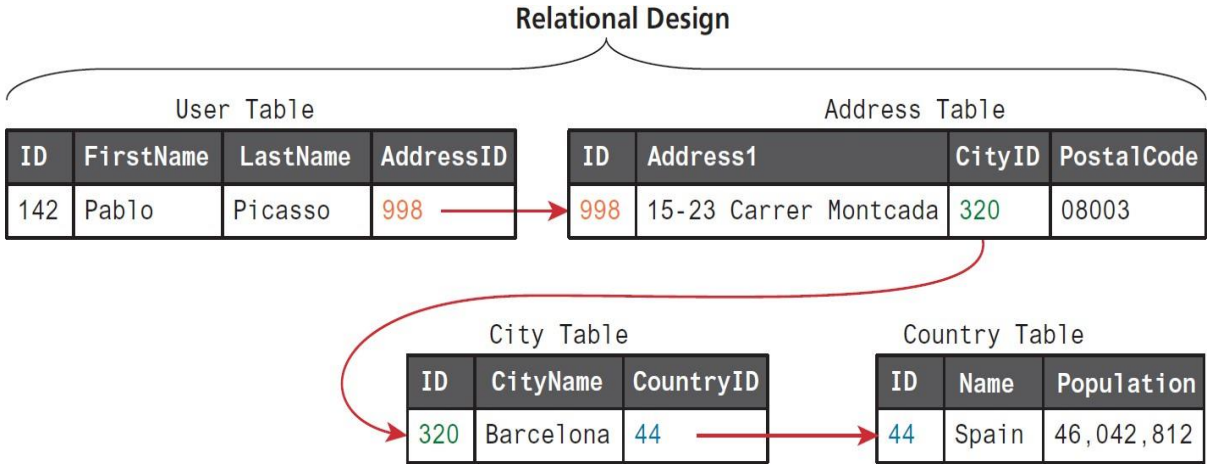
Example of a non-relational database

- JSON and XML document databases.

JSON

```
[
  {
    "name": "John",
    "skills": ["SQL", "C#", "Azure"]
  },
  {
    "name": "Jane",
    "surname": "Doe"
  }
]
```

Relational data versus document store data



Document Store Data

```
{
  "User": {
    "ID": 142,
    "FirstName": "Pablo",
    "LastName": "Picasso",
    "Address": {
      "ID": 998,
      "Address1": "15-23 Carrer Montcada",
      "PostalCode": "08003",
      "City": {
        "ID": 320,
        "CityName": "Barcelona",
        "Country": {
          "ID": 44,
          "Name": "Spain",
          "Population": 46042812
        }
      }
    }
  }
}
```

When to use a relational (SQL) database?

- Structured data where relationship between entities is important: Relational databases require one to follow a schema

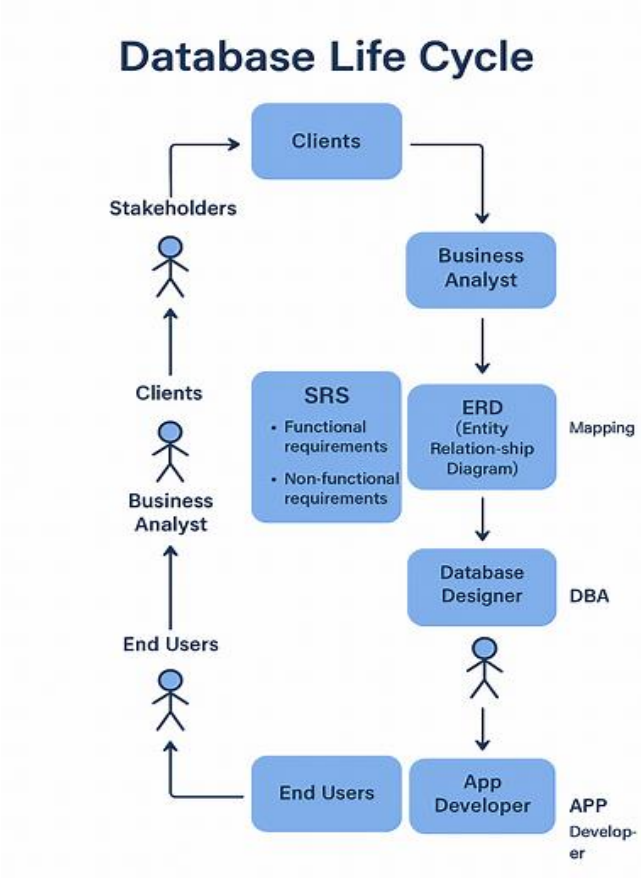
- When to use a non-relational (NoSQL) database
  - If the data you are storing needs to be flexible in terms of shape or size, or if it needs to be open to change in future
  - NoSQL database systems accept some duplication of data

Relational database	NOSQL database
Tables	Collection
Row	Document
Column	Field

- Other comparison criteria: Scalability and data storage, complexity and speed of queries execution, reliability, …

4. Database Life cycle

1. Requirements analysis.
2. Conceptual design (ER diagrams).
3. Logical design (relational schema).
4. Physical design (storage structures).
5. Implementation.
6. Testing & Deployment.
7. Maintenance.



Stakeholders & Clients → Business Analyst → SRS (System Requirement Specification) → Database Designer → ERD (Conceptual Schema) → Logical Schema → DBA (Database Administrator) → App Developer → End Users

5. Main characteristics of the database approach

a. Self-description of nature of a database system

A DBMS catalog stores the description (metadata) of the database. This allow DBMS software to work with different databases.

b. Insulation between programs and data.

Called program-data independence. Allows changing data storage structures and operations without having to change the DBMS access programs (application program)

c. Data abstraction.

A data model is used to hide storage details and presents the users with a conceptual view of the database.

d. Support of multiple views of the data

Each user may see a different view of the database which describes only the data of interset of that user

e. Sharing of data and multiuser transaction processing

Allow a set of concurrent users to retrieve and to update (transaction- concurrency control) the database