DATABASE SYSTEM

1 Chapter I: Introduction to Database

1.1 Some Definitions:

- Database is a collection of related data.
- Data is facts that can be recorded and have implicit meaning.
- Database Management System (DBMS) is a computerized system that enables users to create and maintain a database.
- DBMS is a **general-purpose software system** that faciliates the processes of *defining, constructing, manipulating* and *sharing* databases among various users and applications.

1.2 DBMS System:

- **Defining the database** involves specify the data types, structures, and constraints of the data to be stored in the database.
- **Meta-data**, The database definition or descriptive information is also stored by the DBMS in the form of a database catalog and dictionary.
- **Constructing the database** is the process of storing the data on some storage medium that is controlled by the DBMS.
- **Manipulating the database** includes functions such as querying the database to retrieve specific data, updating the database to reflect changes in the miniworld.
- **Sharing the database** allows multiple users and programs to access the database simultaneously.

1.3 Application Program:

- An **application program** accesses the database by sending queries or requests for data to DBMS.
- A **query** typically causes data to be retrieved.
- A **transaction** causes data to be read and written into the database.

- DBMS also provides functions for **protecting** and **maintaining** the database system:

- **Protecting** includes *system protection* against hardware or software malfuncton (or crash) and *security protection* against unauthorized and malicious access.
- A DBMS need to **maintain** the database system by allowing the system as requirement change overtime.

Database system = database + DBMS

- Conceptual Design.
- Logic Design.
- Physical Design.

- Design of the new application for an existing database or design of a brand new database starts off with a phase called **requirements specification and analysis**.

- These requirements are documented in detail and transformed into a **conceptual design** that can be represented and manipulated by some computerized tools → easily modified, maintained and transformed into **database implementation**.

- The design is then translated into **logical design**. that can be expressed in a data model implemented in a commercial DBMS.

- The final stage is called **physical design**, further specifications are provided for storing and access database.

1.4 Characteristic of the Database Approach:

1. File processing:
    - A traditional **file processing**, each user defines and implements the files needed for the specific software application.
    - Both users are interested in same data but each users maintain separate files and programs to manipulate files.
    - This redundancy in defining and storing data results in wasted storage sapce and in redundant effort to maintain common up-to-date data.
2. Database Approach:
    - In database approach, a single repository maintains data that defined once and then accessed by various users repeatedly though queries, transactions, and application programs.
    - The main characteristic of the database approach vs file processing:
        - Self-describing nature of a database system.
        - Insulation (ngăn cách) between programs and data, and data abstraction.
        - Support of the multiple views of the data.
        - Sharing of data and multiuser transaction processing.

    (a) Self-Describing Nature of a Database System:
    - The database system contain not only the database itself but also a complete definition or description of database structure and constraints.

- The information stored in the DBMS catalog is called **meta-data**, it describles the structure of the primary datatabase.
- NOSQL systems do not require meta-data, those data is sstored in **self-describing data** that includes the data item names and dara values together in 1 structure.
- The DBMS software must work equally well with *any number of database applications*

(b) Insulation between Programs and Data, and Data Abstraction:

- The structure of data files is stored in the DBMS catalog separately from the access program, this is called **program-data indepedence**.
  (Why **meta-data enable data-program independence ?**)
- An operation (known as function or method) is specific in *interface* and *implementation.*
  - interface includes operation name and its data types of its argument.
  - implementation of the operation is specified separately and can be changed without affecting the interface.
- This may be known as **program-operation independence**.
- The characteristic that allows program-data independence and program-operation independence is called **data abstraction**.
- A **data model** is a type of data abstraction that is used to provided this conceptual representation. The data model *hides* storage and implementation out of database users.

(c) Support of multiple views of the Data:

- A multiuser DBMS whose users have a variety of distinct applications must provide facilities for multiple views.

(d) Sharing of Data and Multiuser Transaction Processing:

- This is essential if data for multiple users to be integrated and maintained for a single database.
- The DBMS must include **concurency control** software to ensure that several users trying to update the same data so that the results of update is correct.

3. Actors on the Scene:
   - Database Adminstrators.
   - Database Designers.
   - End users.

1.5 Advantages of Using the DBMS Approach

1. Reduce Redundancy:
   - Data normalization: All logical data item are stored in *only one place* in the database.
2. Restricting Unauthorized Access:
3. Provide persistent storage for Program Object:
4. Provide Storage Structures and Search Techniques for efficient Query processing:
5. Provide Backup and Recovery:
6. Provide Multiple User Interfaces:
7. Represent Complex Relationships among Data:
8. Enforcing Integrity Constraints:
9. Permitting Interfencing and Actions using Rules and Triggers:

1.6 When not to use DBMS

1. It maybe more desirable to develop customized database applications:
   - Simple, well-defined database applications that are not expected to change at all.
   - Stringent, real-time requirements for some application programs that may not met bacase of DBMS overhead.
   - Embedded devices with limited storage capacity, where a general-purpose DBMS would not fit.
   - No multiple user-access data.

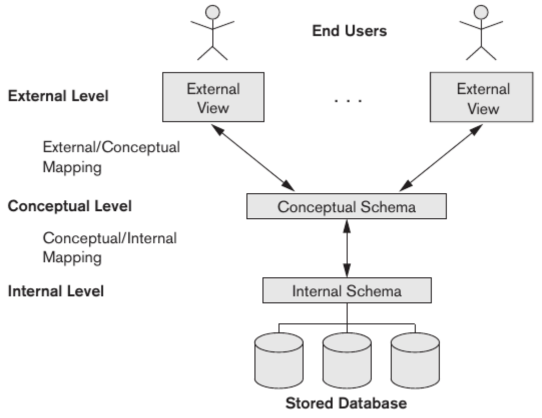
2 Chapter II: Database System Concepts and Architecture

2.1 Data Models, Schemas, and Instances

1. Data Models:
   - A data model is a collection of concepts that can be used to describle the structure of the database - provides the neccessary means to achieve this abstraction.
   - Most data models also include set of **basic operations** for specifying retrievals and updates on the database.
   - Types of Data Model:
     - **High-level** or **conceptual data models** provide concepts that are close to the way many users realize data.
     - **Low-level** or **physical data model** provide concepts that describle the detail of how data is stored in the computer storage media, typically magnetic disk.
     - **Representational** (or **implementation**) **data models** provide concepts that may be easily understood by end users but that are not too far removed (very different from) from the way data is organized in computer storage.
2. Schemas, Instances, and Database State:
   - Schemas is kind of layout of the database.
   - The actual data in the database may change quite frequently. The data in the database at the particular moment of time is called a **database state** or **snapshot**

2.2 Three-Schema Architecture and Data Independence

1. The three-schema Architecture:



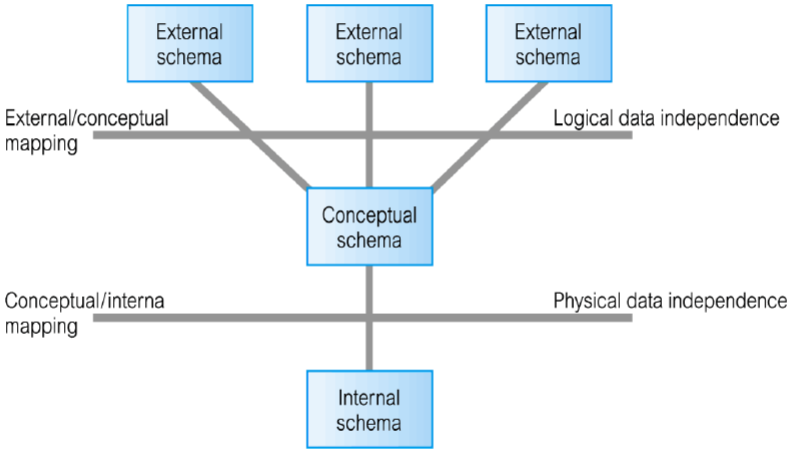
- The **internal level** has an **internal schema** describles the physical storage structure of the database.
- The **conceptual level** has a **conceptual schema** describles the structure of the whole database for users.
- The **external** or **view level** includes a number of **external schemas** or **user views**.

2. Data Independence:

   (a) **Logical data independence:** is the capacity to change the conceptual schema without having to change external schemas or application programs.

   (b) **Physical data independence:** is the capacity to change the internal schema without having to change the conceptual schema

   (c) Summary: Data indepedence occurs when the schema is changed at some level, the schema at the next higher level remains unchanged; only the *mapping* between the teo levels is changed. Hence, aplication referring to the higher level schema do not need to be changed.


External schema
External schema
External schema
External/conceptual mapping
Logical data independence
Conceptual schema
Conceptual/interna mapping
Physical data independence
Internal schema

2.3 Database Languges and Interfaces

1. DBMS Languages:

- **Data definition language** (DDL) : used to define both conceptual and external schemas where no strict separation of levels is maintained.
- **Storage definition language** (SDL):
  - is used to specify the internal schema.
  - DDL is used in conceptual schema.
  - It is used where there is a clear separation between the conceptual and external levels
- **View definition language** (VDL) : is used to specify user views and their mapping to the conceptual and external schemas.
- Beside, DBMS also provides a set of operations or a language called the **data manipulation language** (DML) for manipulating the database.

  - A **high-level** or **nonprocedural** DML.
    - It is used to specify complex database operations concisely (chính xác).
    - Many DBMSs allow high-level DML statements to be entered from the display monitor or terminal or to be embedd in a general-purpose programming language.
    - High-level DML, such as SQL can specify and retrieve many records in a single DML statement, it is called **set-at-a-time**
  - A **low-level** or **procedural** DML.
    - It *must* be embedded in a general-purposes language.
    - It is also called as **record-at-a-time** since it functions related to retrieving and processing individuals records and objects.
- Whenever DMLS commands, whether high-level or low-level, are embedded in a general-purpose language and that language is called the **host language** and DML is called the **data sublanguage**.
- A high-level DML used in a standalone interative manner is called a **query lenguage**.

2. DBMS Interfaces:
   - Menu-based Interfaces for Web Clients or Browsing
   - Apps for Mobile Devices
   - Forms-based Interfaces
   - Graphic User Interfaces
   - Natural Language Interfaces
   - Keyboard-based Database Search
   - Search Input and Output

2.4 The Database System Environment