Chapter (2)
Database Systems Concepts and Architecture

Objectives
Understanding the modern database architectures
Learning some of the terminologies and basic concepts
Learning about data models
Defining the concepts of schemas and instances
Learning about three-schema DBMS architecture
Learning about the typical languages that are provided by a DBMS
Describing the database system software environment
Learning about classification of the types of DBMS packages

Modern DBMS packages are modular in design with a client-server system architecture. In such architectures, the functionality is distributed between client module (application programs and user interface) server module (data storage, access, search, and other functions).

Data Models, Schemas, and Instances

The database approach provides some level of abstraction by hiding details of data storage that are not needed by most database users.

A data model is a collection of concepts that can be used to describe the structure of a database and provides the necessary means and details to achieve this abstraction.

What is the structure of a database:
- the data type
- relationships, and
- constraints
that should be held for all data.

Most data models include a set of basic operations for:
- specifying retrievals (COMPUTE_GPA) and updates
On the database.
Data Models, Schemas, and Instances – cont.

It is also common to include concepts in the data model to specify the dynamic aspect or behavior of a database application.

This allows the database designer to specify a set of valid user-defined operations that are allowed on the database objects.

Examples of user-defined operations:

- COMPUTE_GPA

Generic operations such as insert, delete, modify, or retrieve any kind of object are often included in the basic data model operations.

Categories of Data Models

Data models are categorized into the following groups:

- **High-level or Conceptual data models** – provide concepts that are close to the way most users perceive data,
- **Low-level or Physical data models** – provide concepts that describe the details of how data is stored in the computer. (Computer Scientists)
- **Representational or implementation data models** – provide concepts that may be understood by end users.
- **Object data models** – a new family of higher-level implementation data models that are closer to conceptual data models.

A Conceptual data model uses concepts such as entities, attributes, and relationships.

Entity – represents a real world object or concept
Attribute – represents some property of interest that further describes an entity
Relationship – represents the interaction among the entities.
Determine Entities:

Determine attributes for one of the entities:

Define one relationship:

---

Categories of Data Models – cont.

A representational or implementation data model are used most frequently in traditional commercial DBMS. This model falls between the two extremes the high-level and the low-level data models.

They include relational data model as well as the network and hierarchical models.

The object data model is considered as a high-level data model and is closer to the conceptual data model. We will discuss this in more details in Part III.

The physical data model describes how the data is stored in the computer by representing information such as; record formats, record orderings, and access paths.
Schemas, Instances, and Database State

In any data model, we have to distinguish between the description and the database itself.

The description of a database is called the database schema. This will be defined during the design phase and will not change that often.

A schema diagram is the displayed version of the schema.

Each object on a schema is called a schema construct.

A schema diagram displays only some aspects of a schema such as:
the names of record types and data items, and
some types of constraints.

It does not include:
the data type of each data item,
the relationships among the various files, and
many types of constraints.

Figure 2.1 Schema diagram for the database of Figure 1.2.
Categories of Data Models – cont.

The actual data in a database may change frequently. A data in a particular moment in time is called a database state or snapshot. This is also called the current set of occurrences or instances in the database.

It is very important to distinguish between the database schema and database state.

Let’s see if we can explain the differences. Initially we do not have a database. Once we define a new database, we specify its database schema only to the DBMS. Now we have a database but no data, the database at its empty state. Then, we start inserting data and will populate or load the database. By inserting or deleting data from a database, we will take the database to a new state. The database at any moment in time is in its current state. When we insert data, the DBMS will ensure that the database is in a valid state.

The correct schema is extremely important to DBMS. A rare change in a schema is referred to as the schema evolution.

DBMS Architecture and Data Independence

In previous chapter, we discussed the main characteristics of the database approach; 1) insulation of programs and data, 2) support of multiple user views, and 3) use of a catalog to store the database description (schema).

Here, we will discuss three-schema architecture for database system, which was proposed to help achieve and visualize these characteristics.

The Three-Schema Architecture

The goal of this architecture is to separate the user applications and the physical database.

The schema is defined at the following three levels:

- The internal level
- The conceptual level
- The external or view level.
The **Internal level**

The internal level has an internal schema, which describes the physical storage structure of the database.

The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.
The conceptual level

The conceptual level has a conceptual schema, which describes the structure of the whole database for a community of users.

The conceptual schema hides the details of physical storage structures.

It concentrates on describing:
- entities,
- data types,
- relationships,
- user operations, and
- constraints.

A high-level data model or an implementation data model can be used at this level.

The external or view level

The external level includes a number of external schemas or user views.

Each external schema describes a part of the database that a particular user group is interested in and hides the rest of the database from that user group.

A high-level data model or an implementation data model can be used at this level.

The three schema are only descriptions of data. The only data that actually exists is at the physical level.

In a DBMS database based on the three-schema architecture, each user group refers only to its own external schema. Thus, the DBMS transforms a request specified on an external schema into a request against the conceptual schema, then into a request on the internal schema for processing over the stored database. The requested data in a retrieval process must be reformatted to match the user’s external view. **Mapping** is the process of transforming requests and results between levels.
Data Independence

The three-schema architecture can be used to explain the concept of data independence.

Data independence is the capacity to change the schema at one level of database system without having to change the schema at the next higher level.

There are two types of data independence:

1. Logical data independence
   This is the capacity to change the conceptual schema without having to change external schema or application programs.

2. Physical data independence
   This is the capacity to change the internal schema without having to change the conceptual or external schemas.
Data Independence – cont.

Whenever we have a multiple-level DBMS, its catalog, must be expanded to include information on how to map requests and data among the various levels.

A DBMS uses additional software to refer the required mapping information to the catalog.

Data independence is accomplished because, when the schema is changed at one level, the schema at the next higher level remains unchanged.

The mapping between the two levels is changed to reflect the changes.

The three-schema architecture can make it easier to achieve true data independence both physical and logical. BUT,

The two levels of mapping create an overhead during compilation or execution of a query or program.

Database Languages and Interfaces

A DBMS must provide appropriate languages and interfaces for each category of users.

DBMS Languages
Let’s review. So far:
the design of the database is completed.

Now, the conceptual and internal schema of the database and the required mapping between the two must be completed.

When no strict separation of levels is required, the data definition language (DDL) is used to define both schemas.

The DBMS has a DDL compiler whose function is to process DDL statements in order to identify descriptions of the schema constructs and to store the schema description in the DBMS catalog.
DBMS Languages – cont.

In DBMSs where a clear separation is maintained between the conceptual and internal levels, the DDL is used to specify the conceptual schema only.

To specify the internal schema, the **storage definition language (SDL)** is used.

In this case the mapping can be defined in either one of the two languages.

But wait, for a true three-schema architecture, we need a third language. That language will be used to specify user views and their mapping to the conceptual schema. That language is called the **view definition language (VDL)**.

In most DBMSs, the DDL is used to define both conceptual and external schemas.

To use a populated database, one need a **data manipulation language (DML)** to retrieve, insert, delete, and/or modify data.

---

DBMS Languages – cont.

In current DBMSs, a comprehensive integrated language, such as the Structured Querying Language (SQL), is used that includes constructs for:

- conceptual schema definition,
- view definition, and
- data manipulation.

The storage definition is kept separated so the fine-tuning of the performance will be made easier.

SQL represents a combination of DDL, VDL, and DML, as well as statements for constraint specification and schema evolution.

SQL no longer includes SDL to include conceptual and external levels only.

There are two main types of DML: **high-level (nonprocedural DML)** or **low-level (procedural DML)**.
**DBMS Languages – cont.**

- **A high-level** DML can be used on its own to specify complex operations in a concise manner. SQL is an example.
- **In most DBMSs**, statements are entered interactively from a terminal or by embedding them in a general-purpose code. In the later case, the DML part is identified within the program so that it can be extracted by the compiler.
- **A query in high-level DML** means *which data to retrieve* rather than *how to retrieve it*.

- **A low-level** DML must be embedded in a general-purpose programming language.
- **This type of DML** typically retrieve individual records or objects from the database and process each separately.
- **The low-level DML** needs to use programming language constructs, such as looping, to retrieve and process each record from a set of records.

- Sometimes DML commands, high-level or low-level, are embedded in a **host language** and the DML is called the **data sublanguage**.
- **A stand-alone interactive high-level DML** is called a **query language**.

---

**DBMS Interfaces**

User-friendly interfaces provided by a DBMS may include:

- **Menu-based Interfaces for Browsing**
  Present users with the list of options (menu)

- **Forms-Base Interfaces**
  Displays a form for each use.

- **Graphical User Interfaces**
  Displays a schema to the user in diagrammatic form.

- **Natural Language Interface**
  Accepts requests written in English or some other language and attempts to “understand” them.

- **Interface for Parametric Users**
  Have small set of operations that the user must perform repeatedly, think of the operations by a bank teller.

- **Interface for the DBA**
  Contains privileged commands that can be used only by the DBA’s staff.
The Database System Environment

A DBMS consists of several software components and interacts with several types of computer system software.

**DBMS Component Modules**
- Database System Utilities
- Tools, Application Environments, and Communications Facilities

**DBMS Component Modules**
- The database and the DBMS catalog are usually stored on disk.
- Access to the disk is controlled by the Operating Systems (OS) which schedules disk input/output.
- A higher-level *stored data manager* module of the DBMS controls access to DBMS information that is stored on disk (both in DB and catalog).

*Figure 2.3*  Typical component modules of a DBMS. Dotted lines show accesses that are under the control of the stored data manager.
DBMS Component Modules

The **DDL compiler** processes schema definitions, specified in the DDL, and stores description of the schemas (meta-data) in the DBMS catalog.

The **run-time database processor** handles database accesses at run time. It receives retrieval or update operations and carries them out on the database.

The **query compiler** handles high-level queries that are entered interactively.

The **pre-compiler** extracts DML commands from an application program written in a host programming language.

DBMS System Utilities

Most DBMSs have **database utilities** that help the DBA in managing the database system.

Common utilities have the following types of functions:

- **Loading**: A loading utility is used to load existing data files – such as text files or sequential files – into the database.
- **Backup**: A backup utility creates a backup copy of the database usually by dumping the entire database onto a tape.
- **File reorganization**: This utility can be used to reorganize a database file into a different file organization to improve performance.
- **Performance Monitoring**: A utility to monitor database usage and to provide statistics to the DBA. The statistics is used by the DBA to make decisions such as whether or not to reorganize files to improve performance.
Tools, Applications Environments, and Communications Facilities

There are other tools often available to database designers, users, and DBAs.

**CASE Tools** – are used in the design phase of database systems.

**Data Dictionary (Data Repository) System** – stores information such as: design description, usage standards, application program descriptions, and user information.

**Application Development Environments** – provide an environment for developing database applications and include facilities that help in many faces of database systems, including: database design, GUI development, querying and updating, and application program development.

Tools, Applications Environments, and Communications Facilities – cont.

**Communication Software** – allows users at remote locations to access the database through computer terminals, workstations, or their local personal computers.

The integrated DBMS and data communications system is called a DB/DC system. Often a **Local Area Network (LAN)** is used to distribute DBMSs physically over multiple machines.
Classification of Database Management Systems

Several criteria are normally used to classify DBMSs:

- **Data Model:** DBMS is based on data model. There are two data models used in current commercial DBMSs for this purpose:
  
  - **relational data model**
  - **object data model**

  The relational DBMSs are evolving continuously. This has led to a new class of DBMSs that are called **object-relational DBMSs**.

  Categories of DBMS are based on: relational, object, object-relational, hierarchical, network, and other.

- **Number of Users:** This refers to number of users supported by the systems at a time.


- **Number of Sits:** Locations where the database is distributed.
  
  A DBMS is **centralized** if the data is stored at a single computer site.
  
  A **distributed DBMS (DDBMS)** can have the actual database and DBMS software distributed over many sites, connected by a computer network.
  
  **Homogeneous DDBMS** use the same DBMS software at multiple sites.
  
  **Heterogeneous DBMSs** uses software to access several autonomous preexisting databases stored under **heterogeneous DBMSs**.

- **Cost:** Majority of DBMS packages cost between $10,000 and $100,000. Single-level low-end systems cost between $100 and $3000.
Write an English statements to relate each two boxes that are connected.

Figure 2.4: The schema of Figure 2.1 in the notation of the network data model.