

Chapter 2

Database System Concepts and Architecture

Department of Health Informatics

Aug 2022

Chapter Objectives

By the end of this chapter, you will be able to:

- Differentiate between data models, schemas, and instances
- Explain three schema architecture
- Use selected the database languages and interfaces
- Explain the database system environment
- Distinguish between centralized and server architecture for DBMSs

Presentation Outline

- Introduction
- Data Models, Schemas, and Instances
- Three-Schema Architecture and Data Independence
- Database Languages and Interfaces
- The Database System Environment
- Centralized and Client/Server Architectures for DBMSs
- Classification of Database Management Systems

Introduction

- We are now in the era of distributed DBMS architecture that characterized by client/server modular designs.
- In client/server DBMS architecture, the system functionality is distributed between client and server modules.
- Client module run on a mobile device, user workstation, PC as app programs & user interfaces that access database
- The server module handles data storage, access, search, and other functions.
- In this chapter, various types client/server and basic concepts of database systems architecture, types of interfaces and languages provided by the DBMS and the database system environment are considered.

Data Models

- **Data abstraction** is a process of **hiding unwanted or irrelevant details of data organization** and storage from the end user for better understanding of the data.
- Data abstraction is achieved through a **data model**, a collection of concepts that can be used to describe the **structure of a database (the data types, relationships, and constraints that apply to the data)**.
- Most data models also include a set of **basic operations** to **retrieve, insert, delete, modify** objects on the database.
- We can classify data models as
 - high-level (conceptual),
 - logical (representational or implementational) and
 - low-level (physical) data models.

Data Models...

High-level or conceptual data models provide concepts that are close to the way many users perceive data.

- They use concepts such as **entities (number of tables)**, **attributes (description of the entity; become columns)**, and **relationships**.
- We use **Crow's Foot Symbols** to represent the **relationships**.

A logical or representational data models provide concepts that may be easily **understood by end-users**

- Logical data models hide many details of data storage on disk but can be implemented on a computer system directly.

Data Models...

Features of a logical data model include:

- Includes **all entities and relationships** among them.
- All **attributes** for each entity are **specified**.
- The **PK** for each entity is specified.
 - PK is an attribute that uniquely identifies an instance of the entity.
 - For example, for a student entity, student number is the PK
 - We can have only one primary key in a table.
 - It identifies uniquely every row and it cannot be null.
- **FKs** (keys identifying the relationship between different entities) are specified.
 - FK is a key used to link two tables together.
 - We can have more than one foreign key in a table.
- Normalization occurs at this level.

Data Models...

The steps for designing the logical data model are as follows:

- Specify PK for all entities and find the relationships between entities.
- Find all attributes for each entity.
- Resolve many-to-many relationships.
- Normalization (refine).
- Differences between logical and conceptual data model:
 - In a logical data model PKs are present,
 - In a logical data model, all attributes are specified within an entity.
 - Relationships between entities are specified using PKs and FKs in a logical data model. In a conceptual data model, the relationships are simply stated, not specified.

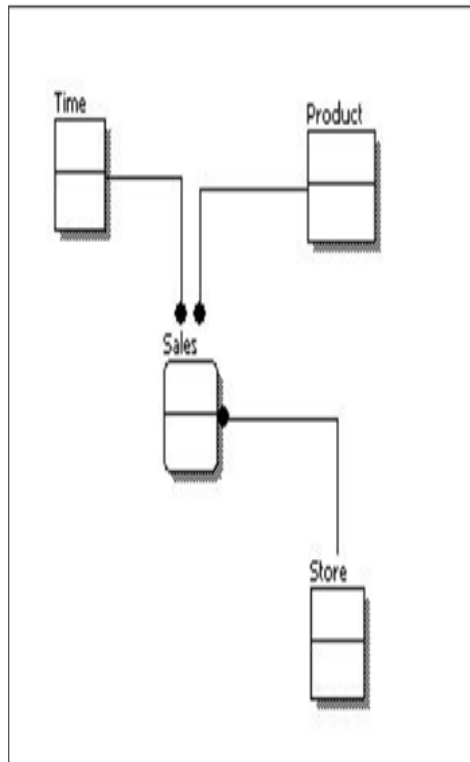
Data Models...

- **Low-level or physical data models** provide concepts that describe the details of how data are **stored on the computer storage media**.
- Concepts provided by physical data models are **incomprehensible for end-users**.
- A physical database model shows **all table structures, including column name, column data type, column constraints, PK, FK, and relationships between tables**.
- Features of a physical data model include:
 - Specification all tables and columns.
 - FK and PK.
 - Denormalization may occur based on user requirements.
 - Physical data model will be different for different RDBMS. For example, data type for a column may be different between MySQL and SQL Server.

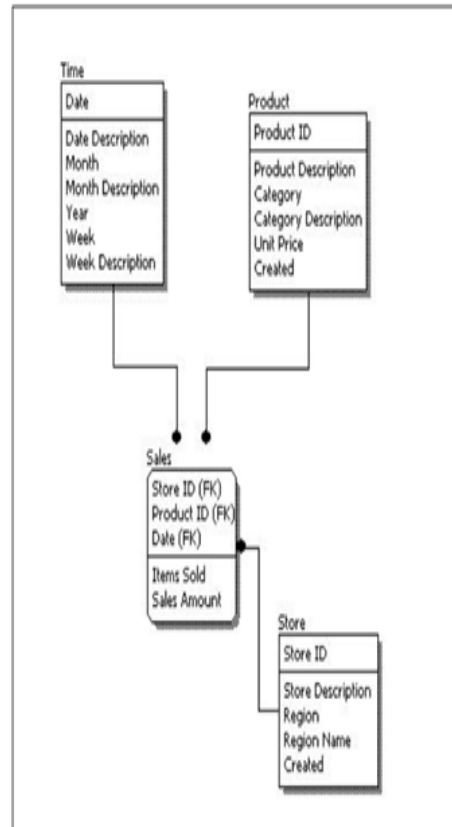
Data Models...

- The steps for physical data model design are as follows:
 - Convert **entities into tables**.
 - Convert **relationships into FKs**.
 - Convert **attributes into columns**.
 - Modify the **physical data model based on physical constraints/requirements**.
- The main differences between the physical and logical data models are:
 - Entity names in the logical model are now **table names**.
 - Attributes in the logical models are now **column names**.
 - **Data type** for each column is specified.

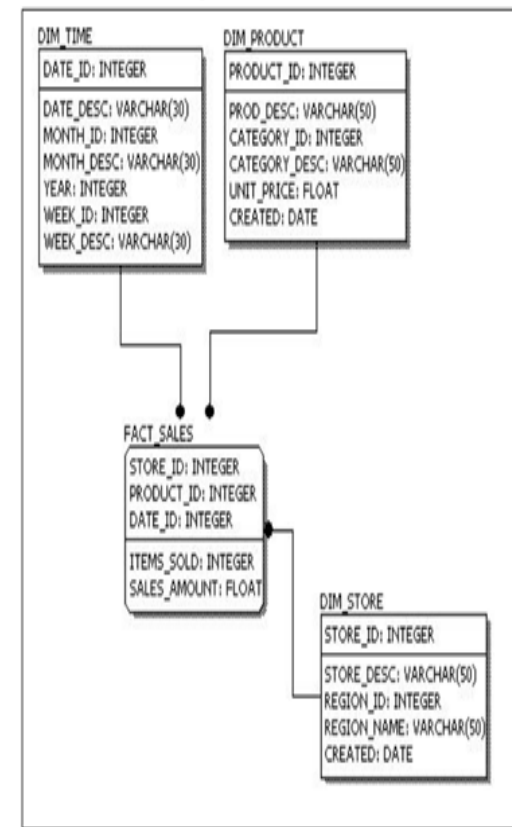
Conceptual Data Model



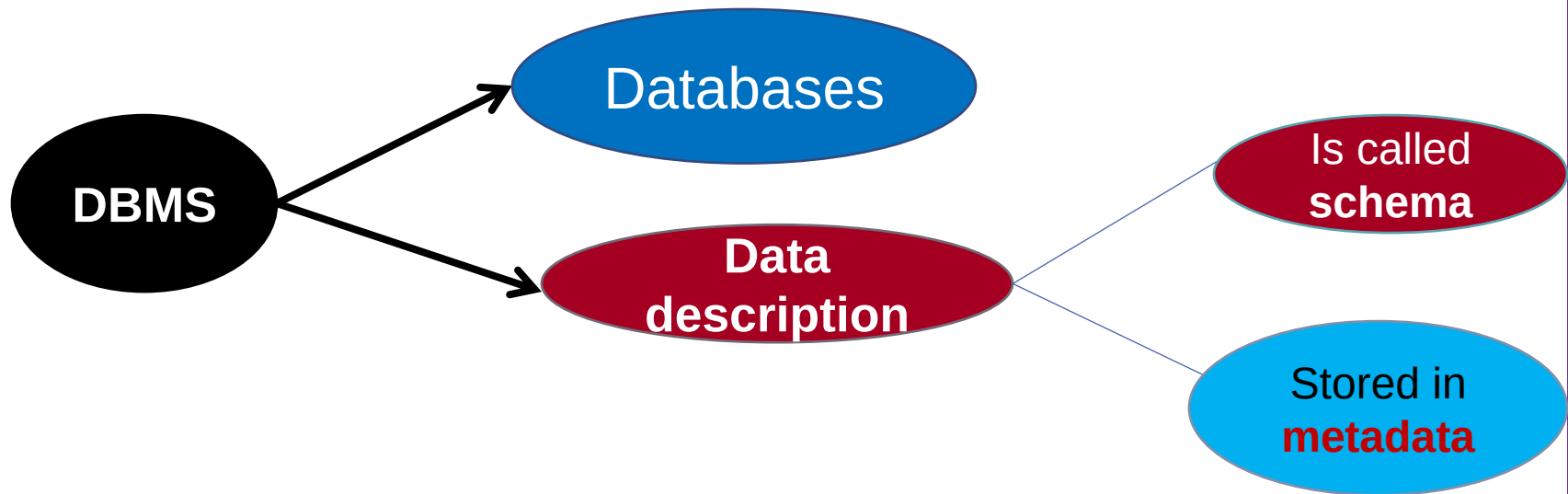
Logical Data Model



Physical Data Model



Data Schemas, Instances, and Database State



Data Schemas, Instances....

- Schemas are **not changed frequently**. If the schema changes are needed, most database systems include operations for allowing such changes.
- The schema defines how data are organized within a relational database; this is inclusive of **logical constraints such as, table names, fields, data types, and the relationships between these entities**.
- Schemas commonly use **visual representations, called schema diagrams**, to communicate the architecture of the database

Data Schemas, Instances....

STUDENT

Name	Student_number	Class	Major
------	----------------	-------	-------

COURSE

Course_name	Course_number	Credit_hours	Department
-------------	---------------	--------------	------------

PREREQUISITE

Course_number	Prerequisite_number
---------------	---------------------

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
--------------------	---------------	----------	------	------------

GRADE_REPORT

Student_number	Section_identifier	Grade
----------------	--------------------	-------

Schema diagram for a hypothetical database

Data Schemas, Instances....

- It is not uncommon that changes occasionally need to be applied to the schema.
- For example, storing another column, called Date_of_birth, for the STUDENT schema above, called **schema evolution**
- Most DBMSs include **operations** for **SE** that can be applied while the database is operational.
- Database schemas are considered the “**blueprint**” of a database describing the **structure of the data, how the data may relate to other tables or other data models.**
- However, the schemas do not show the **actual instances of records.**

Data Schemas, Instances....

- A sample of data from a database at a single moment in time is known as a **database instance or snapshots or a current state or database state**.
- Since database instances are just a snapshot at a given moment, they are **likely to change over time**, unlike database schemas.
- The distinction between **database schema and database state** is very important.
 - When we define a new database, we specify its **database schema** only to the DBMS.
 - Database schema is the **skeleton of database**. It is designed **when the database doesn't exist at all**.

Data Schemas, Instances....

- A database schema **does not contain any data or information**.
- A database instance is a **state of operational database** with data at any given time. It contains a snapshot of the database.
- Database instances tend to change with time.
- A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the **validations**, constraints, and conditions that the database designers have imposed.

Data Schemas, Instances....

- There are three types of database states or instance:
 - **Empty state**: Whenever a new database is defined.
 - **Initial state**: first time data is loaded or populated in database.
 - **Current state**: the present operation is applied to database.
- From the initial state on, every time an **update operation is applied to the database**, we get another database state.
- At any point in time, the database has **a current state**

Data Schemas, Instances....

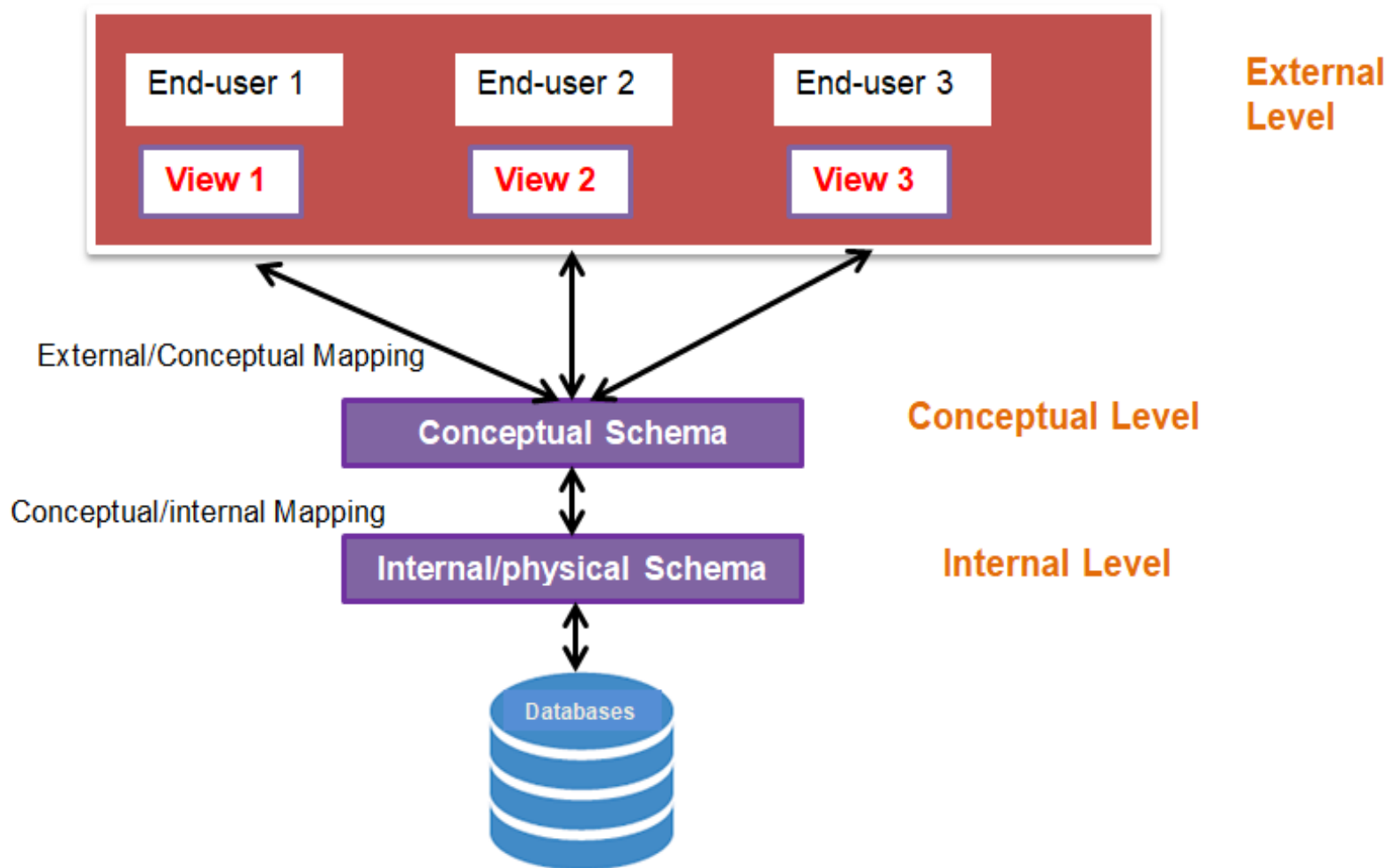
- Key benefits of database schemas include:
 - **Access and security:** Database schema design helps organize data into separate entities, making it easier to share a single schema within another database.
 - **Organization and communication:** Documentation of database schemas allow for more organization and better communication among internal stakeholders.
 - **Integrity:** This organization and communication also helps to ensure data validity. For example, it can help administrators manage normalization processes to avoid data duplication.

Three Schema Architecture of DBMS and Data Independence

- Three of the most important characteristics of the database approach, discussed in the previous chapter are:
 - use of a **metadata to store the schema** (database description) to make it self-describing,
 - insulation of programs and data (**program-data and program-operation independence**), and
 - support of multiple user views.
- The **three-schema architecture** (aka three-level architecture or ANSI/SPARC (American National Standards Institute/ Standards Planning and Requirements Committee) architecture) was proposed to help **achieve and visualize these characteristics**.
- The architecture describes how the data are represented or viewed by the **user** in the database.

Three Schema Architecture ...

- The three **schema architecture** divides the database into three-level to create a **separation between the physical database and the user application**.
- The three schema hides the details of physical storage from the user.
- The database administrator (DBA) should be able to change the structure of database storage without affecting the user's view.
- This architecture contains three layers or levels of the database management system:
 - **External level**
 - **Conceptual or logical level**
 - **Internal level or physical level**



DBMS Architecture

Three Schema Architecture ...

External Schema or View Level

- The highest level of database abstraction.
- Describes the actual view of data that is relevant to the particular user.
- It hides the components that are not relevant to the users.
- Provides different views of the same database for a specific user or a group of users.
- Provides a powerful and flexible security mechanism by hiding the parts of the database from a particular user.

Three Schema Architecture ...

Conceptual or Logical level

- Describes the structure of the whole database.
- Acts as a middle layer between the physical storage and user view.
- Explains what data to be stored in the database, what relationship exists among those data, and what the datatypes are.
- There is only one conceptual schema per database.
- Database administrator and the programmers work at this level.

Three Schema Architecture ...

Conceptual or Logical level...

- This level does not provide any access or storage details but concentrates on the relational model of the database.
- The conceptual schema also includes features that specify the checks to retain integrity and consistency.
- This schema hides the details of the physical storage structure and focuses on describing entities, data types, relationships, constraints, etc.

Three Schema Architecture ...

Internal or Physical level

- The lowest level of database schema.
- Describes the **physical storage** of the database and provides methods to access data from the database.
- It allows viewing the physical representation of the database on the computer system.
- It uses physical data model (abstraction)
- The interface between the conceptual schema and the internal schema identifies how an element in the conceptual schema is stored and how it may be accessed.

Three Schema Architecture ...

- The three-schema architecture is a convenient tool with which the user can visualize the schema levels in a database system.
- Most DBMSs **do not separate the three levels** completely and explicitly, but they support the three-schema architecture to some extent.
- The three schemas are only descriptions of data
- The actual data is stored at the physical level only.
- In the three-schema architecture, each user group refers to its own external schema.

Three Schema Architecture ...

- Hence, the DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database.
- If the request is a database retrieval, the data extracted from the stored database must be reformatted to match the user's external view.
- The processes of transforming requests and results between levels are called mappings.

Data Independence

- The three-schema architecture can be used to further explain the concept of DI.
- DI is the capacity to change the schema at a level of database system without having to change the schema at the next higher level.
- Two types of DI: logical and physical DI.
 - Logical DI: works between conceptual and external schema.
 - It is the capacity to modify the conceptual schema (e.g. adding a new attribute or a new data item) without changing the external schemas or application programs.

Data Independence...

- Physical DI: works between internal schema and conceptual schema.
 - It is the ability to modify the internal schema (e.g. changing the file location or access path, etc.) without changing the conceptual schema. changes may be needed to improve the performance. E.g. changing the access path may been needed to improve the retrieval speed of a particular record.

Data Independence...

- Physical DI exists in most databases where physical details, such as the **location of data on disk**, and **hardware details of storage encoding, placement, compression, splitting, merging of records**, and so on are **hidden from the user**.
- Applications remain unaware of these details.
- Yet, **logical DI is harder to achieve** because it allows structural and constraint changes without affecting application programs—a much stricter requirement.
- DI occurs because when the schema is changed at some level, the schema at the next higher level remains unchanged; only the **mapping between the two levels is changed**.
- Hence, application programs referring to the higher-level schema need not be changed.

Database Languages and Interfaces

- The DBMS provides appropriate languages and interfaces for each category of users
- Database languages, also known as query languages or data query languages, are a classification of programming languages that developers use to define and access databases.
- These languages allow users to complete tasks such as controlling access to data, defining and updating data and searching for information within the DBMS.
- There are different examples of database languages and interfaces, including the following:

Database Languages....

1. Data definition language (DDL)

- Once the design of a database is completed and a DBMS is chosen to implement the database, the first step is to specify **conceptual and internal schemas for the database and any mappings** between the two.
- In many DBMSs where no strict separation of levels is maintained, the DDL is used by the DBA and by database designers to define **both schemas**.
- The DBMS will have 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 (metadata).

Database Languages....

1. Data definition language (DDL)

- For **schema evolution**
- In DBMSs where a clear separation is maintained between the conceptual and internal levels, the DDL is used to specify the conceptual schema only.
- DDL creates the framework of the database by specifying the database schema, which is the structure that represents the organization of data.
- Here's a list of DDL statements in MSSQL:
 - **CREATE**
 - **ALTER**
 - **DROP**
 - **RENAME**

Database Languages....

2. Storage definition language (SDL

- In DBMSs where a clear separation is maintained between the conceptual and internal levels, the DDL is used to specify the conceptual schema only.
- Another language, the storage definition language (SDL), is used to specify the internal schema
- As clear separation between the internal and conceptual schema is not maintained in SQL RDBMSs, they do not have SDL.
- The DDL defines both internal and external schemas in SQL RDMS

Database Languages....

3. Data manipulation language (DML)

- Provides operations that handle user requests, offering a way to access and manipulate the data that users store within a database.
- Here's a list of DML statements:
 - INSERT: Adds new data to the existing database table
 - UPDATE: Changes or updates values in the table
 - DELETE: Removes records or rows from the table
 - SELECT: Retrieves data from the table or multiple tables

Database Languages....

4. Data control language (DCL)

- DCL **controls access** to the data that users store within a database.
- Essentially, this language controls the **rights and permissions** of the database system.
 - **GRANT**: Gives a user access to the database
 - **REVOKE**: Removes a user's access to the database

5. Transaction control language (TCL)

- Manages the **transactions** within a database.
 - **COMMIT**: Carries out a transaction
 - **ROLLBACK**: Restores a transaction if any tasks fail to execute
 - **S**SAVEPOINT****: Sets a point in a transaction to save

Database Languages....

6. Data query language (DQL)

- DQL comprises the main **commands used to fetch data** from a database.
 - **SELECT**: Command to choose/select an attribute from its condition defined by the WHERE condition.
 - **FROM**: Defines the relation for taking input or values for selection in the query.
 - **WHERE**: Defines the conditions to get specific results.

Database Languages....

- In current DBMSs, the above query languages are considered **sublanguages** and are part of a **comprehensive integrated language** (CIL).
- SDL is not included in the CIL and physical storage is defined usually the DBA staff.
- A typical example of a comprehensive database language is the **SQL RDBMS** language which represents a combination of DDL, DQL, DML, DCL and TCL that provide statements for **constraint specification, schema evolution, and many other features**.

Exercise

- Show how to use the 3 database sub-languages (DDL, DML, and DQL) in MSSQL with examples:
- Send the script file to melakuhaile@hu.edu.et
- Due date: 20 Sep 2022

DBMS Interfaces

Menu-Based Interfaces

- Contain the lists of options through which the user is able to send the request.
- a popular technique in **web-based user interfaces**.
- Advantages:
 - provides an easy way to access, manipulate or insert data.
 - No need to remember all the coding or commands.
 - The rollback function is available by pressing back.
- Example: When we search anything in a web browser then it shows all the possible results in a list manner, from which we can click on any option.

DBMS Interfaces...

Form-Based Interfaces

- Display a form to each user.
- The user can fill in the entries to insert new data.
- Example: online form filling for exams.

Apps for Mobile Devices

- These interfaces present mobile users with access to their data.
- For example, banking, reservations, insurance companies, etc. provide apps that allow users to access their data through a mobile phone.

DBMS Interfaces...

Graphical User Interfaces (GUIs)

- All the possible queries are given in an **image format** to make the experience easy and smooth.
- Use the mouse as a pointing device to pick certain parts of the diagram.
- Used by the users who use **electronic gadgets** like mobile phones and touch screens.
- Online games are the best example of it.
- In many cases, **GUIs utilize both menus and forms.**

DBMS Interfaces...

Natural language interfaces

- Accept the request from the user in natural languages (for example, English) and try to interpret it.
- The natural language interfaces have their own schema which is like the database conceptual schema.
- The search engines these days are using natural language interfaces.
- The user can use these search engines to accept the words and retrieve the related information

Speech input and output interfaces

- Accept speech as an input and output the speech as a result.

DBMS Interfaces...

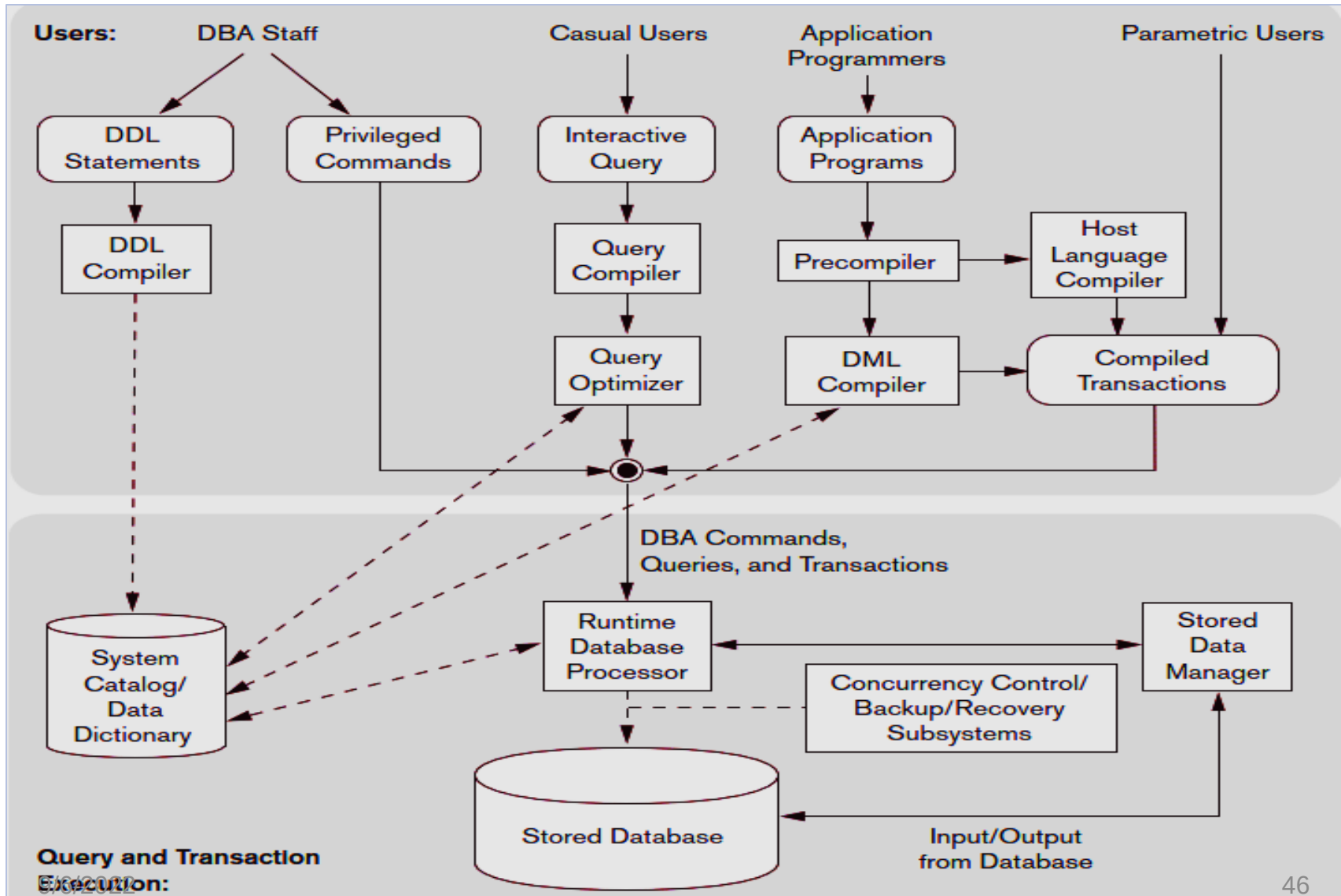
Interfaces for parametric users

- Parametric users like bank tellers have a small set of operations that they must perform repeatedly.
- Contain some commands to perform a request with minimum keystrokes.
- These interfaces can be used in bank transactions to deposit or withdraw money.

Interfaces for DBA

- These interfaces are specially used by database administrators.

The Database System Environment



Database System Utilities

Common DB utilities have the following types of functions:

- **Loading**
- **Backup**
- **Database storage reorganization**: used to reorganize a set of database files into different file organizations, and create new access paths to improve performance.
- **Performance monitoring**: monitors database usage and provides statistics to the DBA.

Centralized and Client/Server Architectures for DBMSs

Centralized DB Architecture (CDBA)

- A CDBA is stored at a single location such as a mainframe computer.
- It is maintained and modified from that location only and is usually accessed using an internet connection
- Were used during the early phases of computing
- Databases were processed remotely and sent to **computer-terminals** power through various communication networks.
- The **terminals had no processing capability** and they were used only to display the information they received.

Centralized and Client/Server....

Client/Server Architecture (CSA)

- Lately, terminals were replaced with PCs, workstations, and mobile devices and CSA replaced CDBA.
- CSA is a computing model in which
 - the **server** hosts, delivers, and manages most of the resources and services to the client.
 - **one or more client computers** (remote processors) connected to a central server (host computer) over **a network or internet connection**.
- Unlike centralized database architectures, CSAs share **computing resources** with clients.

Centralized and Client/Server....

- Clients are situated at **workstations, on PCs on mobile phones**
- Servers are located elsewhere on the network, on more powerful machines.
- Many clients can access the server's information simultaneously
- The RDBMS typically assumes the **client/server architecture**.
 - They process the **query and transaction functionality** and is called **query server or transaction server or SQL Server**.
 - **Open Database Connectivity (ODBC)** provides an **client interface**

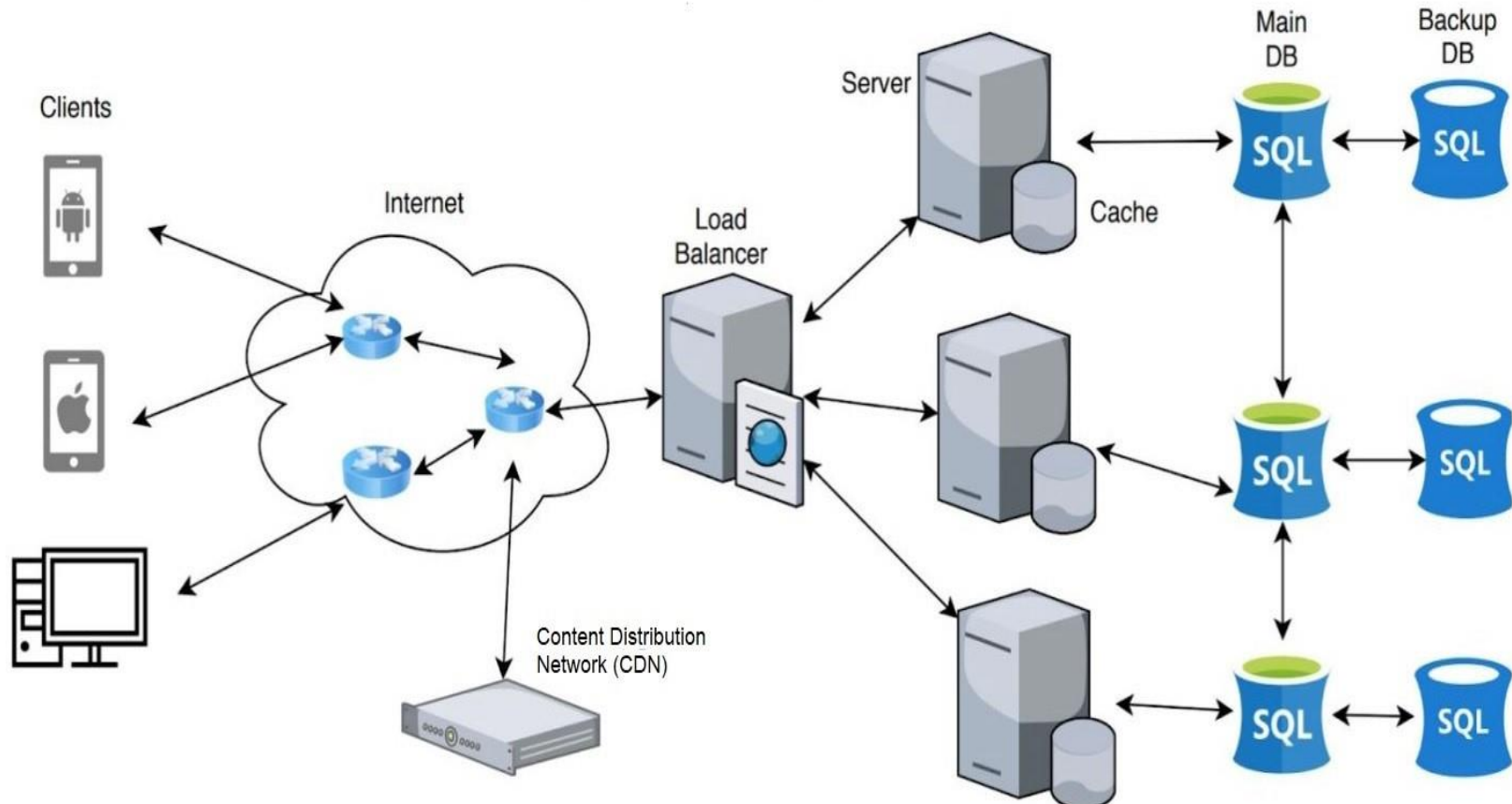


Figure: Client/Server Architecture for DBMS

2 Types of CSA

Two-Tier Client/Server Architecture:

- The **presentation layer** (request inputs from the clients) runs on a client (PC, mobile, tablet, etc.), and
- data are stored on a server called the **second tier**.
- It also provides direct and faster communication.
- An example of 2-tier client/server architecture is the **SQL RDBMS**.
- An example- a **Web server**. It returns the required web pages to the clients that requested them.

2 Types of CSA

Three-Tier Database Architecture

- Is the most popular client/server architecture
- The three-tier architecture contains
 - A **presentation** tier (clients' PCs, workstations, mobile phones, etc.): requests the information
 - The **middle** tier (an application layer (the server)): helps in communication and provides security
 - A **database** server: contains the required data
- Three-tier architectures are Web-based applications
 - Web browsers: presentation layer
 - Middle tier: a development tool for handling a site
 - Database server: the database in server level infrastructure

Classification of Database Management Systems

- A reading assignment.
- Source: Ramez Elmasri & Shamkant B. Navathe. (2016). 2.6. Classification of Database Management Systems. In *Fundamentals of Database Systems* (7th ed., pp. 51–53). New York City, New York: Pearson.