Database System Concepts, Architecture, Languages, Interface for Competitive Exams

Data Models, Schemas and Instances

- A characteristic of the database approach is that it provides a level of data abstraction, by hiding details of data storage that are not needed by most users.
  - A data model is a collection of concepts that can be used to describe the structure of a database. The model provides the necessary means to achieve the abstraction.
  - The structure of a database is characterized by data types, relationships, and constraints that hold for the data. Models also include a set of operations for specifying retrievals and updates.
  - Data models are changing to include concepts to specify the behaviour of the database application. This allows designers to specify a set of user defined operations that are allowed.

Categories of Data Models

- Data models can be categorized in multiple ways.
  - **High level/conceptual data models** – provide concepts close to the way users perceive the data.
  - **Physical data models** – provide concepts that describe the details of how data is stored in the computer. These concepts are generally meant for the specialist, and not the end user.
  - **Representational data models** – provide concepts that may be understood by the end user but not far removed from the way data is organized.
- Conceptual data models use concepts such as entities, attributes and relationships.
  - **Entity** – represents a real world object or concept
  - **Attribute** - represents property of interest that describes an entity, such as name or salary.
  - **Relationships** – among two or more entities, represents an association among two or more entities.

- **Representational data models** are used most frequently in commercial DBMSs. They include relational data models, and legacy models such as network and hierarchical models.
- **Physical data models** describe how data is stored in files by representing record formats, record orderings and access paths.
- **Object data models** – a group of higher level implementation data models closer to conceptual data models.

Schemas, Instances and Database State

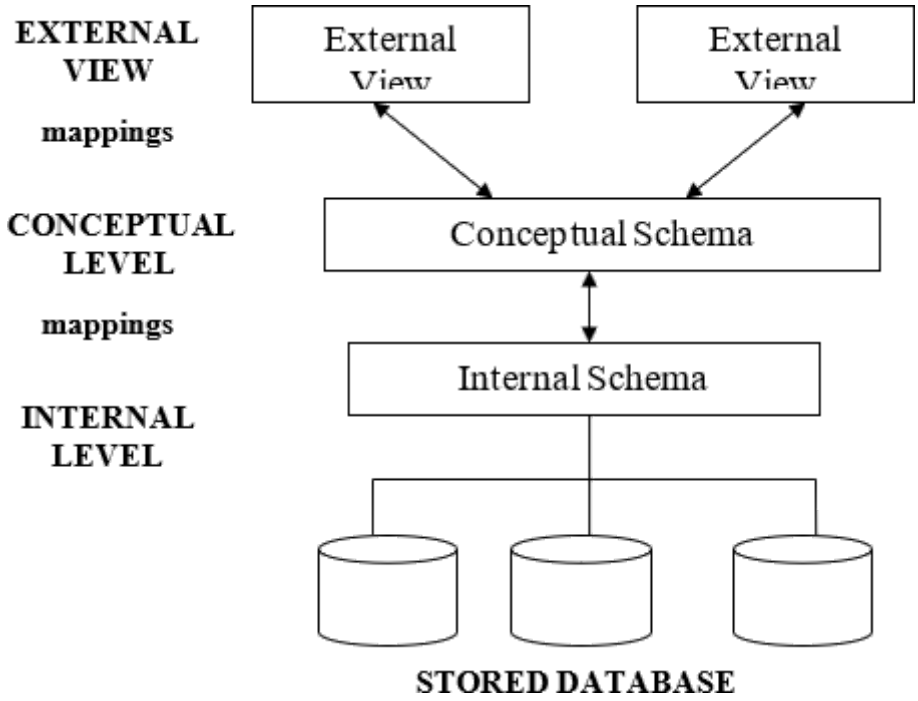
- The description of a database is called the database schema.
  - The schema is specified during database design, and is not expected to change frequently.
  - Data models have conventions for displaying schemas as diagrams. A displayed schema is called a schema diagram.
- Schema Diagram Here
  - Each object in the schema is called a schema construct.
  - Schema diagrams display only some aspects of a schema, such as names and some constraints.
  - The data in a database may change frequently, every time records are added or updated. The data in the database at a given moment in time is called the **database state** or **snapshot**.
- Database Schema vs Database State
  - When a database is defined, the schema is specified to the DBMS. The database state at this point is in the empty state, with no data.
  - The initial state of the database is when the database is first populated or loaded with the initial data. Every time data is added/removed/updated, there is a new database state.
  - The DBMS is responsible for ensuring every state is a **valid state**, a state that satisfies the structure and constraints specified in the schema.
  - The DBMS stores the descriptions of the schema constructs and constraints, called the meta data, in the DBMS catalogue.
  - The schema is called the intension, and the database state an extension of the schema.

Three Schema Architecture and Data Independence

- Remember from the previous chapters, three of the main characteristics of database systems, these are:

- Insulation of programs and data
- Support of multiple views
- Use of a catalogue to store the database description (schema)

- The three schema architecture helps to achieve these characteristics.



Three Schema Architecture

- The goal of the three schema architecture is to separate the user applications and the physical database. The schemas can be defined at the following levels:
  - **The internal level** – has an internal schema which describes the physical storage structure of the database. Uses a physical data model and describes the complete details of data storage and access paths for the database.

- **The conceptual level** – has a conceptual schema which describes the structure of the database for users. It hides the details of the physical storage structures, and concentrates on describing entities, data types, relationships, user operations and constraints. Usually a representational data model is used to describe the conceptual schema.
- **The External or View level** – includes external schemas or user vies. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group. Represented using the representational data model.

- The three schema architecture is used to visualize the schema levels in a database. The three schemas are only descriptions of data, the data only actually exists is at the physical level.
- Each user group refers only to its own external schema. 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 database. The process of transforming requests and results between levels is called mapping.

Data Independence

- The three schema architecture further explains the concept of data independence, the capacity to change the schema at one level without having to change the schema at the next higher level.
  - There are two types of data independence:
    - **Logical data independence**
      - the ability to change the conceptual schema without having to change the external schemas or application programs. When data is added or removed, only the view definition and the mappings need to be changed in the DBMS that support logical data independence.
      - If the conceptual schema undergoes a logical reorganization, application programs that reference the external schema constructs must work as before.
    - **Physical data independence**
      - The ability to change the internal schema without having to change the conceptual schema. By extension, the external schema should not change as well.
      - Physical file reorganization to improve performance (such as creating access structures) results in a change to the internal schema. If the same data as before remains in the database, the conceptual schema should not change.

- For example, providing an access path to improve retrieval speed of section records by semester and year, should not require a query to be changed, although it should become more efficient by utilizing the access path.
- With a multi-level DBMS, the catalogue must be expanded to include information on how to map requests and data among the levels. The DBMS uses additional software to accomplish the mappings.
- Data independence occurs because when the schema is changed at some level, the schema at the next higher level remains unchanged. Only the mapping between the levels is changed.

Database Languages and Interfaces

- Because a database supports a number of user groups, as mentioned previously, the DBMS must have languages and interfaces that support each user group.

DBMS Languages

- DDL – the **data definition language**, used by the DBA and database designers to define the conceptual and internal schemas.
  - The DBMS has a DDL compiler to process DDL statements in order to identify the schema constructs, and to store the description in the catalogue.
  - In databases where there is a separation between the conceptual and internal schemas, DDL is used to specify the conceptual schema, and SDL, **storage definition language**, is used to specify the internal schema.
  - For a true three-schema architecture, VDL, **view definition language**, is used to specify the user views and their mappings to the conceptual schema. But in most DBMSs, the DDL is used to specify both the conceptual schema and the external schemas.
  - Once the schemas are compiled, and the database is populated with data, users need to manipulate the database. Manipulations include retrieval, insertion, deletion and modification.
  - The DBMS provides operations using the DML, **data manipulation language**.
  - In most DBMSs, the VDL, DML and the DML are not considered separate languages, but a comprehensive integrated language for conceptual schema definition, view definition and data manipulation. Storage definition is kept separate to fine-tune the performance, usually done by the DBA staff.
  - An example of a comprehensive language: SQL, which represents a VDL, DDL, DML as well as statements for constraint specification, etc.

Data Manipulation Languages (DMLs)

Two main types:

High-Level/Non Procedural

- Can be used on its own to specify complex database operations.
- DMBSs allow DML statements to be entered interactively from a terminal, or to be embedded in a programming language. If the commands are embedded in a general purpose programming language, the statements must be identified so they can be extracted by a pre-compiler and processed by the DBMS.

Low Level/Procedural

- Must be embedded in a general purpose programming language.
- Typically retrieves individual records or objects from the database and processes each separately.
- Therefore, it needs to use programming language constructs such as loops.
- Low-level DMLs are also called record at a time DMLS because of this.
- High-level DMLs, such as SQL can specify and retrieve many records in a single DML statement, and are called set at a time or set oriented DMLs.
- High-level languages are often called declarative, because the DML often specifies what to retrieve, rather than how to retrieve it.

DML Commands

- When DML commands are embedded in a general-purpose programming language, the programming language is called the host language and the DML is called the data sub-language.
- High-level languages used in a standalone, interactive manner is called a query language.
- Casual end users use high-level query language to specify requests, where programmers usually use embedded DML.
- Parametric end users usually interact with user-friendly interfaces, which can also be used by casual users who don't want to learn the high-level languages.

DBMS Interfaces

- Types of interfaces provided by the DBMS include:

Menu-Based Interfaces for Web Clients or Browsing

- Present users with list of options (menus)
- Lead user through formulation of request
- Query is composed of selection options from menu displayed by system.

Forms-Based Interfaces

- Displays a form to each user.
- User can fill out form to insert new data or fill out only certain entries.
- Designed and programmed for naïve users as interfaces to canned transactions.

Graphical User Interfaces

- Displays a schema to the user in diagram form. The user can specify a query by manipulating the diagram. GUIs use both forms and menus.

Natural Language Interfaces

- Accept requests in written English, or other languages and attempt to understand them.
- Interface has its own schema, and a dictionary of important words. Uses the schema and dictionary to interpret a natural language request.

Interfaces for Parametric Users

- Parametric users have small set of operations they perform.
- Analysts and programmers design and implement a special interface for each class of naïve users.
- Often a small set of commands included to minimize the number of keystrokes required. (i.e.. function keys)

Interfaces for the DBA

- Systems contain privileged commands only for DBA staff.
- Include commands for creating accounts, setting parameters, authorizing accounts, changing the schema, reorganizing the storage structures etc.

Database System Environment

DBMS Components

Stored Data Manager

- The database and the database catalogue are stored on disk
- Access to the disk is handled by the Operating System.
- A higher-level stored data manager controls access to DBMS information that is stored on disk, whether part of the database or the catalogue.
- The stored data manager may use basic OS services for carrying out low-level data transfer, such as handling buffers.
- Once data is in buffers, the other DBMS modules, as well as other application programs can process it.

DDL Compiler

- Processes the schema definitions and stores the descriptions (meta-data) in the catalogue.
- Runtime **Database Processor**
  - Handles database access at runtime.
  - Received retrieval or update operations and carries them out on the database.
  - Access to the disk goes through the stored data manager.

Query Compiler

- Handles high-level queries entered interactively.
- Parses, analyzes and interprets a query, then generates calls to the runtime processor for execution.

Precompiler

- Extracts DML commands from an application program written in a host language.
- Commands are sent to DML compiler for compilation into code for database access. The rest is sent to the host language compiler.

Client Program

- Accesses the DBMS running on a separate computer from the computer on which the database resides. It is called the client computer, and the other is the database server. In some cases a middle level is called the application server.

Database System Utilities

DBMSs have database utilities that help the DBA manage the system. Functions include:

- Loading - used to load existing text/sequential files into the database. Source format and desired target file are specified to the utility, and the utility reformats the data to load into a table.
- Backup – creates a backup copy of the database, usually by dumping database onto tape. Can be used to restore the database in case of failure. Incremental backup can be used which records only the changes since the last backup.
- File Reorganization – reorganize database files into different file organizations to improve performance.
- Performance Monitoring – monitors database usage and provides statistics to the DBA. DBA uses the statistics for decision-making.

Tools, Environments and Communication Facilities

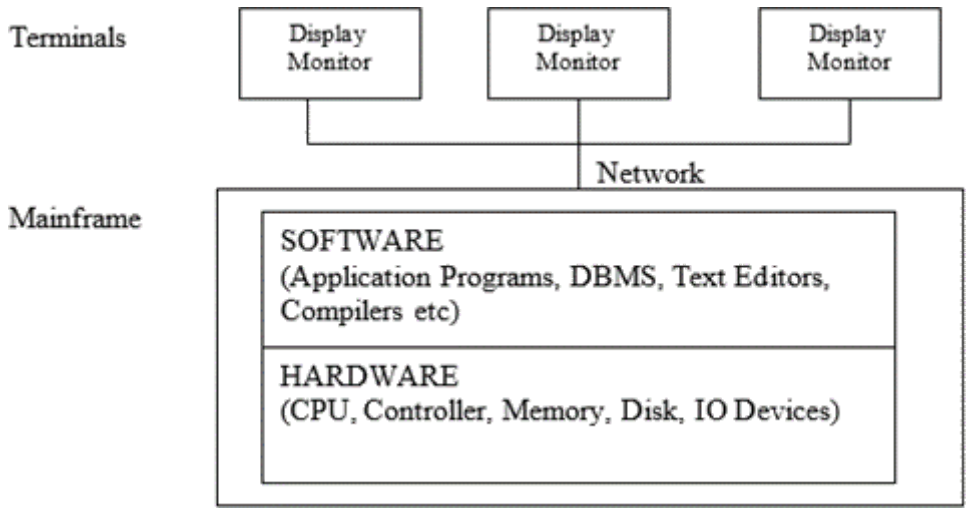
- CASE Tools – used in the design phase to help speed up the development process.

- Data dictionary system – stores catalog information about schemas and constraints, as well as design decisions, usage standards, application program descriptions, user information. Also called an information repository. Can be accesses directly by DBA or users when needed.
- Application development environments – (i.e.. JBuilder) provide environment for developing database applications, and include facilities to help in database design, GUI development, querying and updating and application development.
- Communication software – allow users at remote locations to access the database through computer terminals, workstations or personal computers. Connected to the database through data communications hardware such as phone lines, local area networks etc.

Centralized and Client Server Architectures for DBMSs

Centralized DBMS Architecture

- Used mainframes to provide main processing for user application programs, user interface programs and DBMS functionality
- User accessed systems via 'dumb' computer terminals that only provided display capabilities, with no processing capabilities.
- All processing was performed remotely on the computer system, and only display information was sent to the terminals, connected via a network.
- Dumb terminals were replaces with workstations, which lead to the client/server architecture.



Client Server Architecture

- Define specialized servers with specific functionalities (file servers, print servers, web servers, database servers)
- Many client machines can access resources provided by specialized server.
- Client machines provide user with the appropriate interfaces to utilize servers, as well as with local processing power to run local applications.
- Some machines are client sites, with client software installed and other machines are dedicated servers.
- Client – a user machine that provides user interface capabilities and local processing.
- Server – machine that provides services to client machines such as file access, printing, and database access.

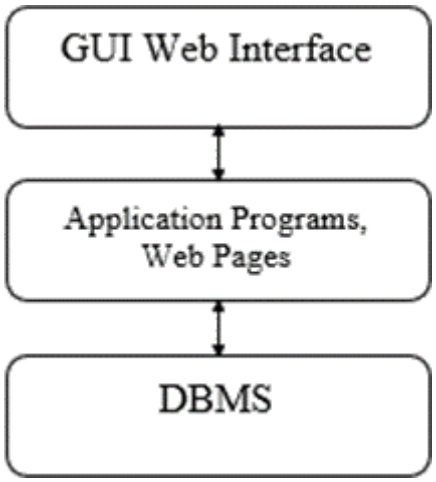
Two Tier Client/Server Architecture for DBMSs

- In relational DBMSs, user interfaces and application programs were first moved to the client side.
- SQL provided a standard language, which was a logical dividing point between client and server.

- Query and transaction functionality remained on server side. In this architecture, the server is called a query server, or transaction server.
- In relational DBMSs, the server is called an SQL server, because most RDBMSs use SQL.
- In such systems, the user interface and application programs run on the client, when DMBS access is needed, the program establishes a connection to the DBMS on the server side. Once the connection is created, the client can communicate with the DBMS.
- ODBC (Open Database Connectivity) is a standard that provides and application processing interface which allows client side programs to call the DBMS as long as both sides have the required software. Most database vendors provide ODBC drivers for their systems.
- Client programs can connect to several RDBMS and send query and transaction requests using the ODBC API
- Query requests are sent from the client to the server, and the server processes the request and sends the result to the client.
- A related Java standard is JDBC, which allows Java programs to access the DBMS through a standard interface.
- These systems are called two tier architectures because the software components are distributed over two systems, the client and server.

Three-Tier Client Server Architecture for Web Applications

- Many web applications use three-tier architecture, which adds an intermediate layer between the client and the database server.
- The middle tier is called the application server, or the web server. Plays an intermediate role, by storing business rules (procedures/constraints) used to access data from database.
- Can improve database security by checking the clients credentials before forwarding request to database server.
- Clients contain GUI interfaces and application specific rules.
- The intermediate server accepts the requests from the client, processes the request and sends the database commands to the db server, then passes the data from the database server to the client, where it may be processes further and filtered.
- The three tiers are: user interface, application rules, and data access.



Classification of DBMSs

- Data Model Classification
  - Relational data model
  - Object data model
  - Hierarchical data model
  - Network data model
  - Object relational data model
- Number of Users
  - Single User systems
  - Multi User systems
- Number of Sites
  - Centralized – data is stored at single site.
  - Distributes – database and DBMS software stored over many sites connected by network
  - Homogeneous – use same DBMS software at multiple sites.
- Cost
  - Low-end systems under $ 3000

- High-end systems, over $ 100,000