DBMS Architecture

A Database Management System (DBMS) is a specialized software designed to help organizations store large amounts of data in an organized way and to make it easy to retrieve, update, and manage that data. It acts as a bridge between users and the physical data stored on disk. Instead of requiring people or applications to deal with raw files, complex file structures, or low-level storage details, the DBMS handles all of these tasks automatically and provides a simple, consistent interface for working with data.

It is important note that a DBMS is not the same thing as a database. The database is the actual collection of data. The DBMS, on the other hand, is the software system that manages this data. A helpful analogy is to think of a library: the books on the shelves represent the database because they contain the information, while the librarians, cataloging system, borrowing procedures, and search tools represent the DBMS because they organize the books, help people find them, protect them, and maintain order. Without the DBMS, the data would still exist, but it would be difficult to manage, easily corrupted, and unsafe to share among many users.

A Database Management System (DBMS) is made up of several internal parts that all work together to store data, manage it, process queries, and ensure everything runs safely. These parts are explained below and depicted in fig. 1.

1. The User Interaction Layer

The topmost level of the DBMS architecture consists of the various users who interact with the system and the different ways in which they send requests. These users include naive users, application programmers, sophisticated users, and database administrators. Naive users interact with the database indirectly through application interfaces, web forms, portals, ATM screens, mobile apps, and other user interfaces. Because they do not understand SQL, their actions are automatically translated into database commands by the application.

Application programmers write the software that naive users rely on. These programs are written in languages such as Java, Python, C#, or PHP and are converted into executable form so that they can communicate with the database system. The resulting application handles user actions, generates the appropriate SQL commands, and sends them to the DBMS for processing.

Sophisticated users, such as analysts or technical staff, interact with the database more directly. They use query tools to write and execute SQL commands without needing a separate application. The database administrator interacts with the database through specialized administration tools. The DBA is responsible for configuring the system, managing storage, setting permissions, enforcing security policies, and monitoring overall database performance.

2. The Query Processor Layer

Beneath the user layer lies the query processor, which functions as the thinking centre of the DBMS. Every request entering the system, passes through this layer. Here, the request is interpreted, analysed, and converted into a set of internal actions the database can execute.

2.1. Application Program Object Code

The first component involved in the request flow is the application program object code. This represents the compiled form of applications created by programmers. Although users interact only with buttons, menus, and forms, the underlying object code quietly generates the SQL commands needed to carry out those actions. When a user fills a registration form or clicks "Search," the program collects the input values and uses them to construct the appropriate SQL statement. This may be an INSERT, UPDATE, DELETE, or SELECT command, depending on the action. The user never sees SQL written anywhere; the object code acts as the translator that turns user actions into precise database instructions and sends them to the DBMS for processing.

2.2. Compiler and Linker

Before the application can produce SQL, the programmer's source code must be converted into a working program. The compiler reads the human-written source code, checks it for errors, and translates it into machine code that the computer can execute. The linker then connects this machine code with the external libraries and database interfaces the program needs in order to communicate with the DBMS. When this process is complete, the final executable becomes the application program object code shown in fig 1. This executable listens for user actions and automatically generates the SQL statements that flow into the rest of the query processor.

2.3. The DML Compiler and Organizer

Once an SQL command arrives in the DBMS, it enters the DML compiler and organizer. This component deals with all commands that manipulate data, such as SELECT, INSERT, UPDATE, and DELETE. It begins by checking that the command is valid according to SQL rules. It then converts this high-level command into an internal form the DBMS can understand. After translating the command, the organizer arranges the steps required to carry it out. It determines which tables will be accessed, the order of operations, and how the data will be filtered or joined. Although the diagram does not show a separate optimizer, the organizing process naturally involves selecting reasonable ways to execute the request efficiently. When this preparation is complete, the DML compiler and organizer hands the instructions to the next component.

2.4. The DDL Interpreter

While the DML compiler handles data-manipulation commands, the DDL interpreter handles commands that define or change the structure of the database. These include statements such as CREATE TABLE, DROP TABLE, and ALTER TABLE. When such a command arrives, the DDL interpreter reads it, understands its meaning, and records the resulting structural information in the data dictionary. The data dictionary is the official reference for the organization of the database, so the DDL interpreter is responsible for maintaining all structural definitions.

2.5. The Query Evaluation Engine

After a command has been translated and arranged, the query evaluation engine performs the actual work. It follows the internal instructions generated by the previous components and interacts with the storage manager to retrieve or update data. It carries out operations such as scanning tables, applying filters, sorting results, joining related records, or modifying existing entries. Once the required data has been processed, it prepares the final output and sends it upward toward the user layer.

3. The Storage Manager Layer

Below the query processor lies the storage manager, which serves as the bridge between the logical commands of the query processor and the physical storage on disk. Because databases typically hold enormous amounts of data, far more than can fit in main memory, the storage manager must carefully arrange data on disk and minimize the number of disk accesses. It is responsible for storing, retrieving, updating, and protecting data.

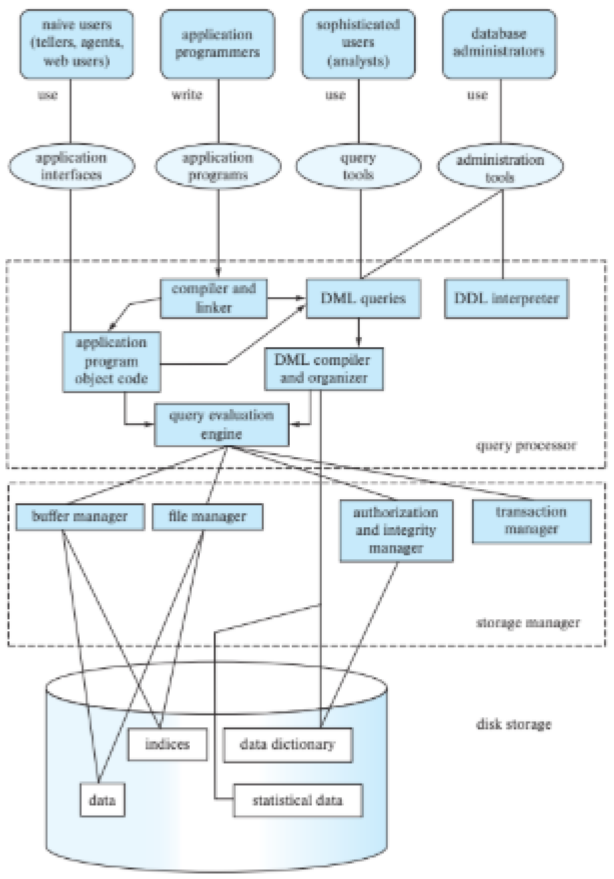

Fig 1: DBMS Architecture

3.1. The File Manager

The file manager handles the lowest-level tasks related to how database files are stored on disk. It decides how to organize database files into blocks and pages, manages the allocation of space, and ensures that the DBMS knows where every piece of data is physically located. The file manager interacts directly with the underlying operating system’s file system, translating database requests into commands that operate on files, blocks, or pages.

3.2. The Buffer Manager

Because reading from disk is slow, the buffer manager plays a crucial role in improving performance. When the query processor needs data, the buffer manager brings the required pages from disk into memory. It also decides which pages to keep in memory and which to remove when space is needed. Effective buffer management is essential because it allows the database to process far more data than could fit into main memory at once. By reducing the number of disk accesses, the buffer manager significantly improves the speed of query execution.

3.3. The Authorization and Integrity Manager

To ensure security and correctness, the storage manager includes the authorization and integrity manager. This component checks whether a user has permission to perform an operation and ensures that all integrity constraints are satisfied. For example, before adding a new row, it verifies whether required fields have values, whether primary keys are unique, or whether foreign keys match existing values. This helps protect the database from unauthorized access and incorrect updates.

3.4. The Transaction Manager

The storage manager also contains the transaction manager, which ensures that the database remains correct and reliable even when many transactions occur simultaneously or when failures occur. The transaction manager enforces the ACID properties; atomicity, consistency, isolation, and durability. It works with logs, recovery routines, and concurrency-control mechanisms to guarantee that a group of actions either all happen or none happen, that crashes do not corrupt data, and that multiple users can work safely at the same time.

4. The Physical Storage Layer

At the bottom of the DBMS architecture is the physical storage layer, where all data is actually stored. This layer contains several types of physical structures. The first is the data files themselves, which hold the actual tables and records. These files contain the values stored in rows and columns for all the tables in the database. Alongside these are the index structures, which serve as quick-lookup tools that allow the DBMS to find data efficiently without scanning entire tables.

Another critical structure in this layer is the data dictionary, which contains metadata about the database. This metadata describes the tables, columns, data types, constraints, user privileges, and storage details. Every time a DDL statement is executed, the DDL interpreter updates this dictionary. Finally, statistical data is stored here. This includes information such as how many rows a table contains or how values are distributed. The query optimizer uses this information to make smart decisions about how to execute queries efficiently. This physical storage layer is the foundation of the entire DBMS. All upper layers depend on it to store, retrieve, and understand the structure of the data.