Syllabus

MODULE-I Introduction [4L] Concept & Overview of DBMS, Data Models, Database Languages, Role of database administrator and database Users, Three Tier architecture of DBMS. Entity-Relationship Model [6L] Basic concepts, Design Issues, Mapping Constraints, Keys, Entity-Relationship Diagram, Weak Entity Sets, Extended E-R features.

Why database?

- **Data** is the raw material for any information system. Data represents a set of characters that have no meaning on their own, i.e., it consists of just symbols. It is very crucial for any system.
- System process the data to transform it into information (meaning is attached to data).
- Data needs to be stored and retrieved efficiently so that efficient process can be developed.

Earlier we used to store the data on **conventional file system** which is a very primitive kind of database management system supported in any OS. It store the data in files. It has following drawbacks:

- **Centralized control** of data is missing. It results followings:
    - Redundancy of data – same data needs to be duplicated in several system
    - Inconsistency of data – It is a corollary of the previous one (One system showing a student belongs to dept1 and another system may show he belongs to dept2)
    - Difficult to enforce standard – entity name, data type, storage policy etc.
    - Difficult to enforce security – access security, backup/restore, corruption etc.
    - Lack of integration – wrong or different data validation rule imposed on same data may results data integration issue. (An employee is paid for 400 hr in a week, while his duty hour per week is 40)
- Data cannot be shared
- Retrieval mechanism is not always efficient
- Sometimes processing logic become complicated, because knowledge of the data organization and access technique is built into the application logic. If we use file system as data store, then application development based on such data store are **data dependent.** If we change the sequential file to index sequential file we need to change the application.
- Difficult to handle concurrency issue (inserting record in a sequential file at a time by two user) and ensure atomicity (all changes take effect, or none do). These result data loss.

To overcome above drawbacks a good integrated Transaction Processing system needs a **Centralized control data store.** Following are the main properties (known as **"ACID test"**) which are enforced in a centralized control data store.

- **Atomicity** - Results of a transaction's execution are either all committed or all rolled back. All changes take effect, or none do. That means, for User's money transfer in a bank system, account balance of both the accounts are adjusted or neither is.

- **Consistency** - The data store is transformed from one valid state to another valid state. Any transaction for changing the data must obey user-defined integrity constraints. For example, suppose that you define a rule that, after a transfer of more than $10,000 out of the country, a row is added to an audit table so that you can prepare a legally required report for the Reserve Bank. Perhaps for performance reasons that audit table is stored on a separate disk. If the audit table's disk is off-line and can't be written, the transaction is aborted.

- **Isolation** - The results of a transaction are invisible to other transactions until the transaction is complete. For example, if you are running an accounting report at the same time that Joe is transferring money, the accounting report program will either see the balances before Joe transferred the money or after, but never the intermediate state where checking has been credited but savings not yet debited.

- **Durability** - Once committed (completed), the results of a transaction are permanent and survive future system and media failures. Say the airline reservation system gives you seat 22A and crashes its disk a millisecond later, it won't have forgotten that you are sitting in 22A and also give it to someone else. It is possible to install a new disk and recover the transactions up to the crashing point, showing that you had seat 22A.

In addition this centralized control data store must also provide following advantage:

- Sharing data in an enterprise
- Enforcing standard on data definition, storage policy etc.
- Apply appropriate security (access security, backup/ restore, data recovery etc.)

Modern database management system can provide all the features needed for developing any complicated enterprise application.

Database

Operational data is basically all data needed for running an enterprise, but it does not include any purely transient information (input or output data, work queue etc.)

A database is a collection of **stored operational data** arranged in a structured form, designed to meet the information requirements of multiple users. It is centrally controlled by the software.

Database Management System (DBMS)

A DBMS is a set of software programs that controls the organization, storage, management, and retrieval of data in a database. DBMS is centrally controlling the DB and any application want to use any data in DB it has to go through DBMS.

The DBMS acts as an interface between the application and the data. It accepts requests for data from an application program and instructs the operating system to transfer the appropriate data.

The following are examples of database applications: computerized library systems , automated teller machines , flight reservation systems , computerized parts inventory systems

Advantages of using DBMS :

- **Data Independence**

  Data independence is a property by which data files are insulated from application programs that use those files. With data independence, a programmer can write programs in FORTRAN, PASCAL, or any other language of his choice without bothering what language programs originally created the files that he wants to access. Data independence can be logical, physical or geographical.

  New categories of data can be added to the database without disruption to the existing system.

  Logical or physical data independence mean the ability to change the logical or physical structures of data without requiring any changes in the application programs which manipulate that data.

  Geographical data independence, a characteristic of distributed database management systems, makes location of data transparent to the users.

- **Data Administration -** It allows organizations to place control of database development in the hands of database administrators (DBAs) and other specialists.

- **Central storage** - It allows different user application programs to easily access the same database.
- **Data Integrity and security** - The degree of correctness, timeliness, and relevance of data stored in a database indicates data integrity. Data integrity can be ensured by certain checks and validation procedures carried out whenever an update operation is attempted and also by the elimination of **data redundancy**. DBMS can enforce integrity constraints. For example in ORACLE (a relational database management system), triggers can be used for this purpose.

- **Access Control** - It can also enforce access controls that govern what data is visible to different classes of user.
- **Reduced Application Development time** – Data access is an important part of an application, which is completely handled by the DBMS. Hence application developer need not develop the detail data access mechanism in each application.
- **Efficient Data Access** - A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently.
- **Concurrent Access and Recovery** – DBMS handles the different issues raised due to concurrent handling of data by various applications so that there is no loss of integrity. It also protects users from the accidental loss due to system failure.

Evolution of DBMS

Second Generation Systems Database Architecture

In this case, the operating system takes care of file management and other routines, thereby relieving the programmer from this effort. Though physical data independence is achieved, logical data independence remains to be built into the system.

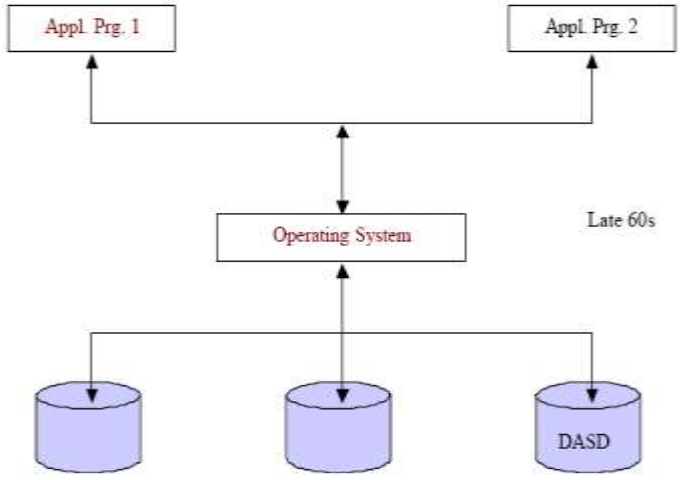


Fig. 1.3b : Second generation systems

Third Generation Systems

Here a layer of DBMS has been added over the operating system. The systems provide physical as well as logical data independence.

Now most of the system follows **three-tier architecture** where DBMS is residing in a separate tier. Presentation logic and application logic consist of other two tires. Application tire always access the store data using DBMS.

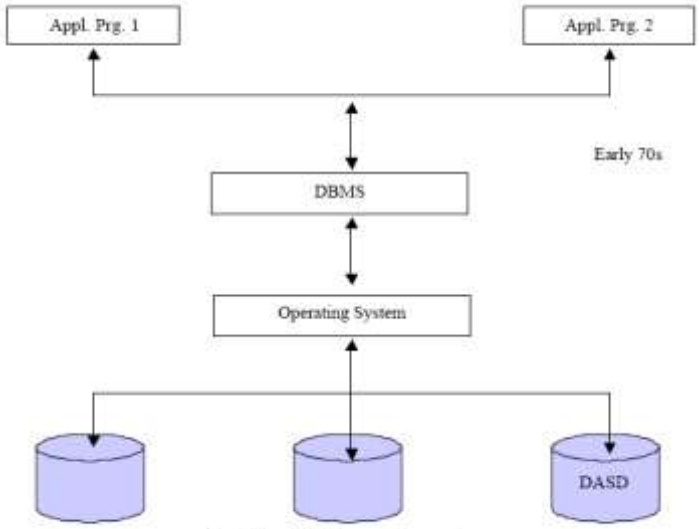


Fig. 1.3c : Third generation systems

Different Types of DBMS

Based on the internal data organization, DBMS can be classified mainly as **Relational**, **Network**, and **hierarchical**. RDBMS is widely used.

A relational database management system (RDBMS) is a database management system (DBMS) that is based on the **relational model** as introduced by E. F. Codd. Most popular commercial and open source databases currently in use are based on the relational model. In RDBMS data is stored in the form of tables and the relationship among the data is also stored in the form of tables.

At a minimum, RDBMS must satisfy followings:

- Presented the data to the user as relations (logical view to user)
- Provided relational operators to manipulate the data in tabular form
- Data can be accessed or reassembled in many different ways without having to reorganize the database tables.

RDBMS also have following features :

- The **structured query language (SQL)** is used to retrieve and manage data in RDBMS.
- It is relatively easy to create and access and easy to extend.
- The definition of a relational database results in a table of metadata or formal descriptions of the tables, columns, domains, and constraints.

Some example DBMS products : IBM's DB2, Oracle Database, Microsoft SQL Server, PostgreSQL, MySQL, and many others. Most of these use the SQL data definition and query language.

Database Languages

SQL (structured query language) is a standard 4GL language for interfacing with relational DBMS. It is a language used to retrieve/modify and management of data in RDBMS by following categories of applications :

- Any application developed in different high level languages
- SQL forms, Report writers etc.
- System Utilities used by system manager to take back-up/restore of databases, load data into a database and other jobs related to database administration.

The function of a RDBMS can be divided into two essential parts :

- **Data definition part** – It provides definition or description of database objects and is written using **data definition language** (DDL). This part creates logical structures of entities when a database is set up, for example the fields and sub fields to be created, the length, type, etc.

- **Data manipulation part** - (retrieval, addition, deletion, modification) refers to methods of manipulating data and is implemented using **data manipulation languages** (DML).

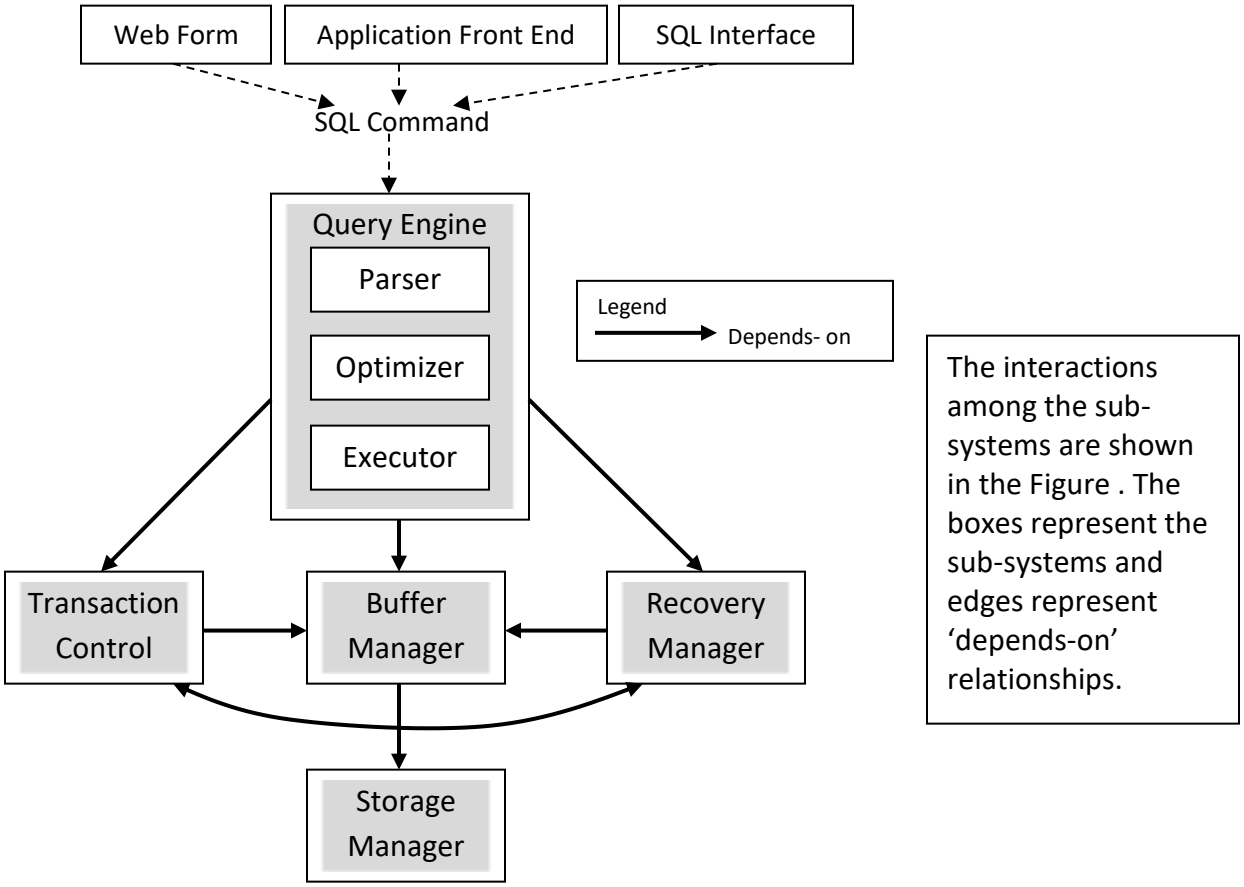
Components of DBMS

The **DBMS** accepts SQL commands generated from a variety of user interfaces, produces query evaluation plans, executes these plans against the database, and returns the result. Typical DBMS has a layered architecture.

The DBMS lies between the users of the database. It depends on the underlying operating system's file system to access the database files and also for memory space and resource management.

A simplified version of (MySql) a DBMS structure is shown below. It consists of following main components.

The Query Engine receives all the SQL commands from the users, parses the SQL, optimizes them, prepares a plan of execution, executes the plan and sends the user the appropriate response. We can define this sub-system as the front end of any DBMS.



The query engine depends on the buffer manager to read or write data to satisfy a user query. It also depends on transaction control to manage transaction and control concurrency. The transaction control in turn depends on the buffer manager to store copies of modified data temporarily for the lifetime of a transaction. It also depends on the recovery manager log modifications made in the data. The buffer manager only depends on the storage manager to store or retrieve data to or from the file system. The recovery manager depends on the storage manager to store backup data in the file system. It depends on the buffer manager to re-apply the modifications while recovering the data during crash recovery. Finally it depends on the transaction manager to recover from any active transactions during the time of crash.

The query engine sub-system consists of a Parser, Optimizer and Executor. The parser, optimizer and executor depend on each other sequentially to serve a user query.

- The parser checks the syntax of SQL commands, checks the existence of requested tables and columns, and checks user privileges to perform the operation.
- The optimizer checks the existence of indexes to use to satisfy the query, chooses among several alternate ways to execute the command and prepares an optimal plan of execution.
- The executor takes the actions required to execute the plan or requests other components of the system to take necessary actions.

The **Buffer Manager** lies in between the query engine and storage manager. This sub-system is responsible for memory management. It performs caching of data managed by the storage manager for fast retrieval. The buffer manager supplies the requested data to the query engine. It maintains a data buffer for table and index file, a metadata buffer for data dictionary files and redo buffer for the log files and the undo buffer, on behalf of the transaction manager.

The **Storage Manager** on the other hand is the backend of the system. This sub-system deals with persistent and efficient storage and retrieval of data in the OS file systems.

The **transaction control** sub-system consists of Transaction Manager and Lock Manager components. The components of the **recovery** manager are the Logger and Backup & Recovery. The logger records any modifications done in the database in log files. Using the backup & recovery component, a DBA can save copies of the data files and later use them in conjunction with log files to restore the database in a consistent state at the point of time just before the crash happened.

Reading from a table :

The user passes a SQL statement to the parser. The parse checks the syntax of the query, and checks with the buffer manager to ensure the existence of the queried table/column in the metadata, and proper user authorization levels. A parse tree is then constructed out of the query. The tree is passed to the optimizer which checks for the availability and appropriateness of indices, finds alternate execution plans, and picks the best one. The executor then executes the plan, reads the data that satisfies the query from the data buffer, and performs the appropriate relational algebraic operations.

If the buffer manager does not find the required data in its buffers, the appropriate data file is checked (via the storage manager), and the data is obtained, buffered, and returned.

Writing to a Table:

The user passes a SQL statement to the parser requesting an update of data. The parse checks the syntax of the query, checks with the buffer manager to ensure the existence of the table/column in where to write, and proper user authorization levels. A parse tree is then constructed out of the query. The tree is passed to the optimizer which checks for the availability and appropriateness of indices,

finds alternate execution plans, and picks the best one. The executor then executes the plan by requesting the transaction manager to handle the write.

Before actually writing the data transaction manager does couple of things. It checks if the operation will violate any business rules defined in the metadata, it asks the lock manager to put a lock on the records to be modified, keep a copy of the data in the undo buffer, and then modify data in the data buffer. Transaction manager also informs the logger to record the write operation. The logger stores this information in the redo buffer, which is ultimately written into the log file. Beginning and completion of transactions are also recorded in the log.

If the data to be modified in not in the buffer, the buffer manager obtains the data from storage manager, places it in the buffer and then performs the modification. However, if the data to be modified is currently being modified by another user transaction, the write either has to abort or has to wait for the other transaction to end.

Architecture of a DBMS

- For efficient retrieval of data a complex data structure is used for representation of data in the database.
- To hide this complexity from common database users three levels of abstraction are defined by which the database may be viewed.
- They are **logical or external** view; **conceptual** view; and **internal or physical** view.
- Schema, level and view are used interchangeably to describe the architecture of a DBMS. This architecture is also sometimes referred to as **ANSI/SPARC** (American National Standards Institutions/ Standards Planning and Requirements Committee) architecture. The three schemas (levels) in the architecture are only descriptions of data.

- A **model** is an **abstract description** of something more complex (such as the reality)
- An **abstraction** means "omitting details" of some complex system that are usually not important in certain situation.

Schema

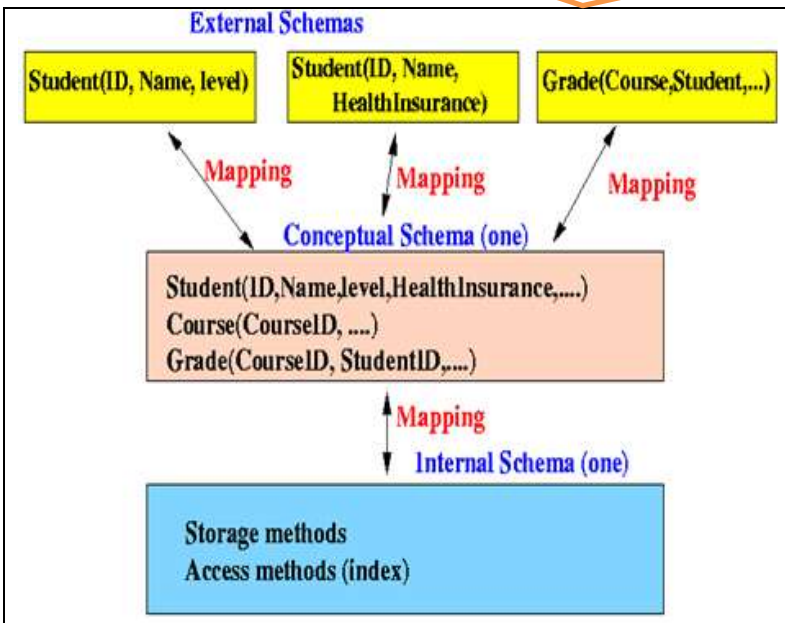
The **database schema** contains the **definitions/descriptions** that enable the **database system** to satisfy the needs of **all its users** . The database schema is specified during the **design of the database** and is **not expected to change frequently**.

A **schema** is an overall conceptual or logical view of collection of all the tables in a database. The schema contains a list of all the fields, the field types, the maximum and minimum acceptable values for each field, along with information about the structure of every row in every table of the database. A **schema** describes the view at each of these levels.

The 3 Schema Architecture in Database Systems

A **Database System** must accommodate **the information need** of **many different kinds of users** . To accommodate this large number of users that **use the same database system** , the 3 different types of schemas (= description of the database) are used. They are independent of each other i.e. each description can be extended without disturbing other description.

Figure shows an example of the 3 levels of data description of a DBMS



In most DBMSs that support user views, external schemas are specified in the same data model that describes the conceptual-level information.

- **External scheme** is for end user (non-experts). We use SQL views to describe this. It is also sometimes called subschema.

- **Conceptual schema** is used by the designer of the database to specify how the data will be stored inside the DBMS.

The Conceptual schema describes the various fields in data files without giving details on how the data files are stored.

It is the most important data description in a database system and must be designed with care. It is written using the DDL (Data Definition Language).

- **Internal/Physical schema** : The Physical schema describe how the data file are stored inside the computer and how to BEST access the data files (e.g., through indices)
  The Physical schema includes things like: File storage structures, Keys, Indices, etc...