Sagi Rama Krishnam Raju Engineering College (Autonomous)

Department of Information Technology

Database Management Systems

(DBMS)

Academic Year: 2022-23. Class: 2/4 B Tech (IT) 1st Semester. (R-20)

UNIT-1

Introduction to DBMS

Contents:

-Database system

- Characteristics (Database Vs File System)

- Database Users (Actors on Scene, Workers behind the scene)

- Advantages of Data base systems

- Database applications.

- Levels of Abstraction Offered by DBMS

-Brief introduction of different Data Models

- Concepts of Schema, Instance and data independence

- Three tier schema architecture for data independence

- Database system structure, environment

-Centralized and Client Server architecture for the database

**Database**: A database is a structured collection of useful and meaningful data which describe the activities of an organization.

**Ex 1**: The database of a University contains data about students, courses, faculty, departments, classes, results etc which describe a set of entities and relationships among those entities.

**Ex 2**: The database of a Hospital contains data describing doctors, specialities, patients, facilities, diagnostics, laboratories, pharmacy etc.

**Database Management System**: A DBMS is software or a set of programs designed to store, maintain and retrieve data efficiently from a database. A DBMS:

- Allows users to create their own database.

- Acts as interface between data and application programs.

- Maintains security of data while allowing application programs to retrieve the data.

**Ex**:

* MySQL
* Microsoft Access
* Oracle
* PostgreSQL
* dBASE
* FoxPro
* IBM DB2 etc.

**Characteristics** (Database System Vs File System):

Consider the following example. A university maintains data about all professors, students, departments and course offerings. Let the following application programs are designed to maintain the data.

-Adding new professors, students, courses.

-Enrolment of students to courses.

-Assigning grades and computing GPA for each student.

New application programs are added to the system as and when needed.

Prior to the development of DBMSs, data is maintained in the form of operating system files which has the following drawbacks.

-**Redundancy and inconsistency**: Various files created by different programmers are likely to have different formats. Same data may be repeated in multiple files which lead to ***redundancy*** which in turn leads to higher storage and access cost. For example, if a student takes two courses from two departments, his address may be stored in both the departments. Later, if the address is changed, the change may be reflected in one department file leaving the other unchanged. This problem is called ***inconsistency***.

-**Difficulty in accessing data**: As the application programs are developed for specific tasks only, sometimes it is difficult to get the needed data from the files. For example, the university may want to generate the list of students who got GPA above 8.0. Suppose that there is an application program that generates a list of all students along with GPAs. Now, the university has to trim the list manually to get the list of 8.0 GPA students or it may want to develop a new application program that generates list of students with GPA above 8.0.

-**Data Isolation**: As the data is scattered in multiple files and different files are likely to be in different formats, writing new application programs to retrieve appropriate data scattered in multiple files will be difficult.

-**Integrity Problems**: Generally, data values stored must follow certain constraints. For example, the GPA of a student should not exceed 10 and the grade for a course should fall in range of ‘A’ to ‘F’. When writing an application program, appropriate code can be included to meet such constraints. However, adding new constraints to an existing program may be difficult.

-**Atomicity Problems**: A computer system may subject to failure for several reasons. In such cases, the data must be restored to a consistent state that existed before failure. Certain operations must be ***atomic*** in nature, i.e. they must happen in entirety or not at all. For example, updating individual course grades and updating GPA should be done as an atomic unit. Else, only grades will be updated without updating the resulting GPA. Ensuring atomicity will be difficult in file processing system.

-**Concurrent Access Anomalies**: Generally, computer systems allow concurrent access of data by several users for the sake of getting faster response. Sometimes, this may leads to incorrect final state of data. For example, the number of registrations made by two clerks for some course may give wrong final count if they are done concurrently.

-**Security Problems**: Certain parts of data may need to be protected from unauthorized usage. For this purpose, the system must be capable of securing different parts of data at different levels. This sort of security can’t be provided by conventional file system.

A DBMS provide solutions for all problems listed above.

**DBMS vs. Flat File System:**

|  |  |
| --- | --- |
| **DBMS** | **Flat File Management System** |
| It Supports Multi-user access. | It does not support multi-user access. |
| Designed to fulfil the need for small and large businesses. | It is only limited to smaller DBMS system. |
| Removes redundancy and maintains Integrity. | Redundancy and Integrity issues present. |
| Expensive. But in the long term Total Cost of Ownership is cheap. | It's cheaper. |
| Easy to implement complicated transactions. | No support for complicated transactions. |

**Database Users (Actors on Scene, Workers behind the scene)**

A variety of people are concerned with creation and use of databases. They are:

**Naïve Users**: These are unsophisticated users who interact with the system by invoking one of the application programs that are already written. For example, a clerk in a bank invokes an application program to debit or credit an account, fills in the necessary fields and submit.

**Application Programmers**: These are computer professionals who write application programs. They can develop user interfaces by choosing tools like RAD( Rapid Application Development). Using such tools, forms and reports can be constructed with minimum effort.

**Sophisticated Users**: These users interact with the system without writing application programs. They form their requests using a query language.

**Specialized Users**: These are sophisticated users who write specialized database applications that do not fit into traditional data processing frame work. Examples include computer aided design systems, knowledge base and expert systems, environment modelling systems etc.

**Database Administrator** (DBA): A major reason behind use of DBMS is to have centralized control over data and also the programs that access the data. Database Administrator is a person who has such central control. The functions of DBA include:

**Schema Definition**: The DBA creates the database schema by executing a set of data definition statements. DBA creates both physical and logical schema by interacting with the users to know their requirements.

**Schema and physical organization modification**: The DBA carries out changes to schema and physical organization to meet the changing needs of users and to improve performance.

**Granting authorization for data access**: DBA grants different privileges to different types of users to control their access to sensitive parts of data. Users can access data in permitted ways only. The authorization details are maintained in a special system structure.

**Routine Maintenance**: DBA periodically takes back up onto remote servers to prevent loss of data in case of any unexpected circumstances. Also, ensures that free disk space is available for normal operations and upgrades disk space as needed. DBA monitors jobs running on the database and ensure that system performance is not degraded by expensive tasks.

**Advantages of Data base systems**:

* **Data Independence**: Application programmes are never exposed to details of data representation and storage. This is achieved through three levels of abstraction offered by DBMS. This makes the task of programmers easier.
* **Efficient Data Access**: DBMS utilizes a variety of techniques and powerful functions to store and retrieve data efficiently. This feature offers benefit when the data are stored on external devices.
* **Data Integrity and Security**: DBMS can enforce integrity constraints to maintain data semantically and as per organization requirements. DBMS also offers control on data access by granting privileges to various classes of users.
* **Data Administration**: DBMS serves as an efficient handler to balance the needs of multiple applications using the same data. It adopts uniform administration procedures for data. Centralizing data administration results in efficient retrievals.
* **Concurrent Access and Crash Recovery**: A DBMS schedules concurrent access to the data in such a manner that only one user can access the same data at a time for writing purpose. In addition, DBMS maintains enough structures to recover the system to normal state in the event of any crash.
* **Reduced Application Development Time**: As the DBMS supports many important functions that are common to many data access applications. New applications can be developed quickly and with little efforts.

**Database applications**: Few of the database system applications include the following:

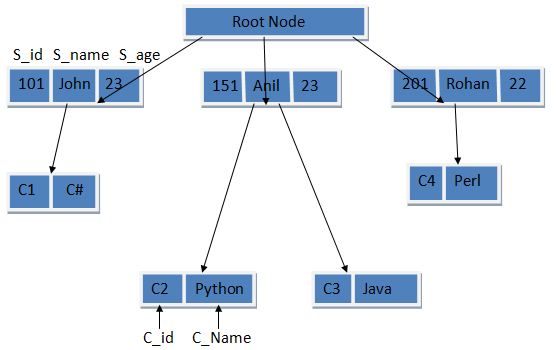
* **Telecom**: There is a database to keeps track of the information regarding calls made, network usage, customer details etc. Without the database systems it is hard to maintain that huge amount of data that keeps updating every millisecond.
* **Industry**: Where it is a manufacturing unit, warehouse or distribution centre, each one needs a database to keep the records of ins and outs. For example distribution centre should keep a track of the product units that supplied into the centre as well as the products that got delivered out from the distribution centre on each day; this is where DBMS comes into picture.
* **Banking System**: For storing customer info, tracking day to day credit and debit transactions, generating bank statements etc. All this work has been done with the help of Database management systems.
* **Sales**: To store customer information, production information and invoice details.
* **Airlines**: To travel though airlines, we make early reservations, this reservation information along with flight schedule is stored in database.
* **Education Sector**: Database systems are frequently used in schools and colleges to store and retrieve the data regarding student details, staff details, course details, exam details, payroll data, attendance details, fees details etc. There is a hell lot amount of inter-related data that needs to be stored and retrieved in an efficient manner.
* **Online Shopping**: You must be aware of the online shopping websites such as Amazon, Flipkart etc. These sites store the product information, your addresses and preferences, credit details and provide you the relevant list of products based on your query. All this involves a Database management system.

**Brief introduction of different Data Models**:

A DBMS allows users to define database in terms of a data model. Every DBMS supports a specific data model. A ***data model*** is a collection of high level data description constructs that hide many low level storage and implementation details. For example, **Oracle** supports relational data model.

A data model hides many details from users and it is closer to how DBMS stores the data rather than how users think of underlying organization. Hence, we need a model that reflects users view more closely. A ***Semantic data model*** is a more abstract, high level data model that facilitates users to come up with a good initial description of activities of the organization. Semantic models are not supported by most of the DBMSs. Hence, a semantic model should be translated into the data model supported by chosen DBMS. ***Entity - Relationship model*** is a widely used semantic data model that allows pictorial representation of the database.

**Types of data models**: Various data models supported by different DBMSs are:

i) **Hierarchical Model:** In a Hierarchical database model, data is organized in a tree-like structure. Data is Stored in hierarchical (top down or bottom up) format. Data is represented using parent-child relationship. In Hierarchical model, one parent may have many children, but one child has only one parent. 

ii) **Network Model:**

The network database model allows each child to have multiple parents and each parent to have multiple children. It helps to address the need to model more complex relationships like many-to-many relationship. In this model, entities are organized in a graph which can be accessed through several paths. Boxes represent entities and lines represent relationships.

Teacher1

Course1

Teacher2

Course2

Teacher3

Course3

Student3

Student2

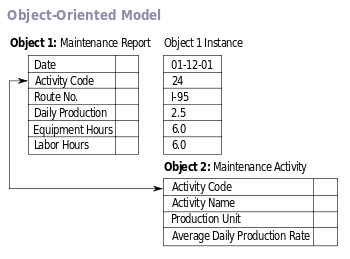
Student1

iii) **Relational Model:**

Relational model is the most widely used DBMS model because it is one of the easiest. This model allows users to represent database as a collection of tables each with an instance. Data stored in relational model can be normalised to remove redundancy and other anomalies. Relational model stored in fixed structures and manipulated using SQL.

iv) **Object Oriented Model:**

In Object-Oriented model, data is stored in the form of objects. The structure which is called classes display data within it. It defines a database as a collection of objects which stores data members, values and operations that are allowed on the data. Inter relationships and constraints are implemented through objects and links. This model can handle more complex applications such as scientific experiments, geographic information systems etc.



**Concepts of Schema, Instance and data independence**:

**Schema and Instance**: The description of database in terms of a data model is called ***Schema***. A database schema specifies the overall design of the database. The design of database at physical level is called ***Physical Schema***. The design of database at logical level is called ***Logical Schema***. The design of various views corresponds to ***External Schemas***.

A database contains a set of data at given moment. The data in a database change from time to time in terms of volume and data values. The collection of information stored in the database at a given moment is called an ***Instance***. The instance of a database corresponds to the values of variables in a program at a point during execution of the program.

**Data Independence**: Data independence is an important advantage of DBMS. Application programs are not exposed to the changes in the way data is structured and stored. This property is achieved through three levels of abstraction. Usually, view relations are generated on demand from the relations corresponding to conceptual level. If the relations in conceptual level are changed, we can redefine the view so as to get the same result as before. That is, users at external level are insulated from changes made at conceptual level. This is called ***Logical Data Independence***. Similarly, sometimes physical storage structures need to be changed for various reasons like improving retrieval performance. Users at logical level are not exposed to such changes made. This property is called ***Physical Data Independence***.

**Three tier schema architecture for data independence (Levels of abstraction)**: A database system allows users to access and modify data through a set of programs. DBMS provides users with an abstract view of data. That is, the system hides from the users, the details of how data is stored and maintained internally.

**….** Defines

View 2

View 1

View n

views for

various

user

groups.

Defines **what** data

Logical Level

has been stored.

Physical Level

Defines **how** data

has been stored.

A database system stores huge volumes of complex data. For the system to be usable, it must retrieve data efficiently. For this purpose, complex data structures are used to represent the data. Most of the users of database systems are not computer trained. Hence, to simplify the interaction of users with the system, DBMS offers three levels of ***abstraction***.

i) **Physical Level**: This level defines ***how*** data are actually stored on the disk. This level describes complex low level data structures in detail.

ii) **Logical Level**: This level, also called **Conceptual Level** describes ***what*** data are stored in the database. This level defines database in terms of simple structures. i.e., it defines a collection of tables to represent the database.

iii)**View Level**: This level, also called **External Level** is the highest level of abstraction and describes several views of the underlying database each intended for a specific group of users. Each view is exposed to a subset of the database in which a group of users are interested.

**Ex**: Consider a University database storing details of Students, Courses, Faculty, and Departments etc. Physical level for this database defines what data structures to use to store the data, creation of indexes on various data parts to speed up the retrievals etc. Logical level defines what tables to create to represent the data. It also includes specifying various integrity constraints and semantics. At the view level, one view for this database is to display a list of courses along with faculty and count of enrolments. Additional views are defined as needed.

**Database system structure, environment**:

**Structure of DBMS**: Figure 1.5 shows a simplified structure of a DBMS based on relational model. The DBMS accepts commands from various user interfaces. It then produces query evaluation plans, executes the plans on the database and returns the results.

When a user issues a query, the query is ***parsed*** and presented to optimizer. The ***optimizer*** uses the data storage information to produce an efficient execution plan. An ***execution plan*** is a strategy to be followed and is represented as a tree of relational operators.

The query evaluation engine interacts with the ***Files and Access Methods*** layer. This layer supports the concept of a file, which is a collection of pages or records. This layer supports indexes and organizes the information within a page. ***Buffer Manager*** brings relevant pages from disk to main memory and then back from memory to disk when finished. ***Disk Space Manager*** deals with managing space on the disk. Higher layers read and write pages through this layer.

The DBMS supports **concurrent execution** of transactions with careful scheduling to see that anomalies do not occur. It also maintains a log of all changes to the database. ***Transaction Manager*** ensures that transactions request and release locks according to some protocol. The ***Lock Manager*** keeps track of lock requests and grants locks on database objects when available.

The ***Recovery Manager*** maintains a log of all changes done by various transactions to the database. In the event of crash, this log and other structures maintained during normal execution are used to restore the system to normal state.

The disk contains actual data. The disk also maintains Meta Data, i.e. data about data. Index files are also maintained on the disk.

Unsophisticated users (Customers, Travel Agents etc ) Sophisticated Users, Application Programmers, DB Administrators.

Application Front Ends

SQL Interface

Web Forms

SQL COMMANDS

Shows Command Flow

Shows

Interaction

Query

Evaluation

Engine

Concurrency Control Manager **DBMS**

Parser

Plan Executor

Optimizer

Operator Evaluator

Files and Access Methods

Recovery Manager

Transaction Manager

Buffer Manager

Lock Manager

Disk Space Manager

**Index Files**

**System**

**Catalogue**

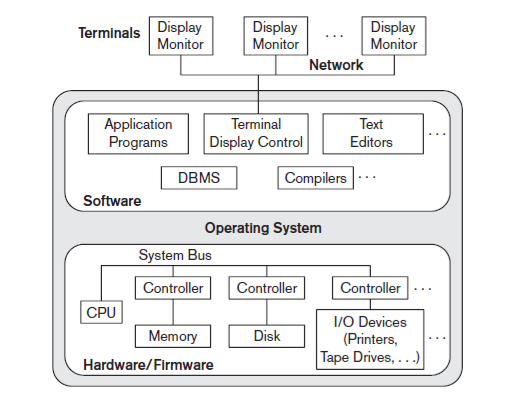
**Data Files**

Shows References

**DATABASE**

Centralized and Client Server architecture for the database:

In Centralized Architecture, a single computer system is used for processing all system functions including User application Programs and User Interface Programs as well as DBMS functionalities. Therefore the Processing used to take place in these Computer Systems and the display information is sent to display terminals and these terminals are connected to mainframe computers via various kinds of Networks.



Centralized Architecture for DBMS

## Client/Server Architecture:

A client in this framework is typically a user machine that provides user interface capabilities and local processing. When a client requires access to additional functionality, such as database access which does not exist at that machine, it connects to a server that provides the needed functionality. A server is a system which contains both Hardware and Software which provides services to client machines like file access, printing and database access. There are two models in Client/server architectures.

**1) Two Tier Client/Server Architecture for DBMS :**

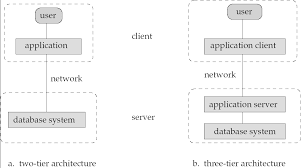
Here Two-tier means that architecture has two layers, which are client layer and data layer. Client layer have several Client machines which can have the access to the database server. The API present on the client machine will establish the connection between the machine and the Database server through JDBC or something else. This is because Clients and Database Server may be at different locations. Once this connection gets established, the Interface present on the client machine contains an Application Program on the back-side which contains a query. This query will be processed by the Database server and in turn the queried information will be sent to the client machine.

For example, if we query the database to retrieve some information, the query will be processed by Database server and that information will be sent to the client by Database server itself.

**2) Three-Tier client/server Architecture for DBMS:**

Here there is an additional layer which acts as an intermediary between Client layer and Data layer called Business logic layer. Business logic layer is the layer where the Application Programs are processed. Here the Application Programs are processed in the Application server itself, which makes it different from Two-tier Architecture where queries are processed in the database server.

Simply the Client machines will contact Application Server which in turn processes our Application Programs and fetches the Required Data from Database and then sends this Information back to the client machine in the suitable format only.



Three-Tier client/server Architecture for DBMS

It may appear that Two-Tier Architecture is easy to use and maintain. But, Two-tier architecture is not scalable when we have large number of clients and also not secure because the clients are having direct access to database server. But Three-Tier Architecture ensures Scalability and Security of the data because of the presence of this Intermediate layer which processes the queries and it just retrieves data from server instead of processing in the server to take place.

**When not to use DBMS**?

A DBMS is complex software optimized for certain kinds of workloads. Sometimes, using a DBMS may not be suggestible for certain kinds of applications. Following are the reasons.

- Its performance may not be adequate for certain specialized applications. For example, applications with tight real time constraints or applications containing few well defined customized critical operations.

- An application may need to manipulate data in ways that are not supported by a query language. For example, relational databases do not support analysis of text data.

-The hardware and software cost of DBMS is high and hence small organizations do not prefer the use of databases.

**--------XXXXXXX--------**