Module-1 : Introduction to DBMS

Overview :

Data : Data is distinct pieces of information formatted in a special way. Data can exist in a variety of forms as numbers or text on pieces of paper, as bits and bytes stored in electronic memory, or as facts stored in a person's mind.

Information : When data is processed, organized, structured or presented in a given context so as to make it useful, it is called information.

Example : The average score of a class

(or)

~~Info~~

The entire school as information that can be derived from the given data.

Database : A database is a collection of data, typically describing the activities of one or more related organizations.

For Example : ~~A~~ University database might contain information about the following :

1. Entities such as students, faculty, courses and classrooms
2. Relationships between entities, such as students enrollment in course, faculty teaching courses and the use of classrooms for courses.

Database Management System (DBMS):-

A DBMS is a collection of interrelated data and a set of Programs to access those data.

→ A database system is simply a computerized record-keeping system. The database can be regarded as a container for a collection of computerized data file.

- → Adding new, empty files to the database
- → Inserting data into from existing files
- → Retrieving data from existing files
- → Changing data in existing files
- → Deleting data from existing files
- → Removing existing files from the database

History:

The first general purpose DBMS, designed by Charles Bachman at general electronic in the early 1960's was called the integrated data store. It formed the basis for the network data model, which was standardized by the conference on data system languages and strongly influenced database systems through the 1960's. Bachman was the first recipient of ACM's Turing Award (The Computer science equivalent of a Nobel Prize) for work in the database he received the award in 1973.

In the late 1960's, IBM developed the information

management system (IMS) DBMS, used even today in many major installations.

In 1970's Edgar codd, at IBM's San Jose Research Laboratory proposed a new data representation framework called the relational data model.

In the 1980's the relational model consolidated its position as the dominant DBMS paradigm and database systems continued to gain widespread use. SQL query Language for relational databases, developed as part of IBM's. SQL (Structured Query Language) 1999 was adopted by the American National standard Institute (ANSI) and International Organization for Standardization (ISO).

DBMS Applications:

DBMS are used by many individuals either directly or indirectly. Some of the applications of DBMS are listed below.

1. Banking: The user accesses the bank database for creating or debiting the account. The bank database stores the details of individual customers, their account, loans and banking transactions.
2. Airlines: For schedule information and for reserving the ticket, we access the

database of airlines. The database should be reliable.

3. Student at University:- For student information, course registration and grades.
4. Credit and Transactions:- For purchases on credit cards and generating of monthly statements.
5. Telecommunication: For keeping records of calls made, generating monthly bills, maintaining balances in prepaid calling cards, and storing information about the communication network.
6. Finance: For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds.
7. Sales: For customer, Product and purchase information.
8. Library: This database stores details of the book available. The user access this database to find a book quickly. The database allows easy management by allowing a user to reserve a book and intimate him through a mail when the book is available. The system also sends a reminder to customers who did not return the book by due-date. The system uses a bar code. Reader to Provide access to database.
9. Super Market: The Cashier places the bar code of a product against a barcode reader. The application Program uses the bar code to identify the price

and reduces the number of items on the database.

10. Human Resources:- For information about employee, salaries, payroll taxes and benefit and for generation of paychecks.

File System Vs DBMS:-

Introduction to File System: One way to keep information on a computer is to store it in, permanent system i.e, files.

→ Files is a memory block of a secondary storage device like disk, magnetic-tape etc., to store logically related records permanently.

→ Therefore, in a file management system, each record has separate file. Various records are stored in various files and programs are written to work with different files respectively.

Keeping organization information in a file-processing system has a number of major disadvantages.

Disadvantages:

1. Data Redundancy and Inconsistency: As various copies of same data resides, in several files indicating the redundancy (waste copies) of data, which leads to higher storage and access cost.

2. Difficulty in Access Data: To access the appropriate data, each time the related new request program has to be written or the needed information is extracted manually.

3. Data Isolation: Because data scattered in various files, and files may be in different formats, writing new application program to retrieve the appropriate data is difficult.

4. Integrity Problems: The data values stored in the database must satisfy certain types of consistency constraints. To develop the new consistent range in the existing system, an appropriate code must be added in various application problems.

5. Atomicity Problems: A computer system, like any other mechanical or electrical device, is subject to failure. In many applications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existed prior to the failure.

6. Concurrent Access Anomalies: For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously.

7. Security Problems: Not every users of the database system should be able to access all the data.

Advantages of DBMS:-

Using a DBMS to manage data has many advantages

1. Data Independence: Application programs should not, ideally, be exposed to details of data representation and storage, The DBMS provides an abstract view of the data that hides such details.
2. Efficient Data Access: A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently. This feature is especially important if the data is stored on external storage devices.
3. Data Integrity and Security: To develop the new consistent range in the existing system, an appropriate code must be added in one application progoam that accers all data at one time.
As a security property, every user of DBMS is able to access only needed data and strictly not allowing to access all the data.
4. Data Administration: when several users share the data, centralizing the administration of data can offer significant improvements.
Experienced professionals who to understand the nature of the data being managed, and can be responsible for organizing the data representation to minimize redundancy and for fine-turning the storage of the data to make retrieval efficient.

5. Concurrent Access and Crash Recovery: A DBMS schedules concurrent access to the data in such a manner that users can think of the data as being accessed by only one user at a time. Further, the DBMS protects users from the effects of system failures.

6. Reduced Application Development Time: The DBMS supports important functions that are common to many applications accessing data in the DBMS. The high level interface to the data, facilitates quick application development.

Storage Data:

Database is a collecting of large volume of data. A user will not always require the complete data. The responsibility of the database system is to provide only the data that is required to the user.

→ A data Model is a collection of high-level data description constructs that hide many low-level storage details.

→ A DBMS allows a user to define the data to be stored in terms of a data model.

→ Most database management systems today are based on the relational data model.

→ A Semantic data Model is a more abstract, high-level data model that makes it easier for a user to come up with a good initial description of the data in an enterprise.

→ A DBMS design in terms of a semantic model serves as a useful starting point and is subsequently translated into a database design in terms of the data model the DBMS actually supports.

→ The widely used semantic data model called the entity-relationship (ER) model allows us to pictorially denote entities and the relationships among them.

1. The Relational Model:-

The central data description construct in this model is a relation, which can be thought of as a set of records.

→ A description of data in terms of a data model is called a schema. In the relational model, the schema for a relation specifies its name, the name of each field (or attribute or column), and the type of each field.

Example:

Student information in a university database may be stored in a relation with the following schema:

Students(sid: string, sname: string, login: string, age: integer, cgpa: real)

The preceding schema says that each record in the students relation has five fields, with field names and types as indicated.

An example instance of the students relation appears in below table:

Sid	Sname	login	age	cgpa
53666	Jones	jones@cs	18	3.4
53688	Vasu	vasu@pt	18	3.2
53650	Smith	smith@math	19	3.8
53631	Smith	smith@ee	11	1.8
53632	Ravi	Ravi@music	12	2.0

Table: An Instance of the students Relation.

→ Each row in the students relation is a record that describes a student.

→ Every row follows the schema of the students relation. The schema can therefore be regarded as a template for describing a student.

→ Every student has a unique sid value. observe that to capture this information by simply adding another field to the students schema.

→ The ability to specify uniqueness of the values in a field increases the accuracy with which we can describe our data.

→ So specifying integrity constraints is an important aspect of a data model.

Other Data Models:

Hierarchical Model (e.g: used in IBM's IMS DBMS)
Network model (e.g: used in IDS and IDMS)
object-oriented Model (e.g used in objectstore & versant)
object-relational Model(e.g used in DBMS Products)

2. Levels of Abstraction in a DBMS:-

The data in a DBMS is described at three levels of abstraction, as shown in below diagram. The database description consists of a schema at each of these three levels of abstraction:

- The conceptual
- physical
- External.

The data definition Language (DDL) is used to define the external and conceptual schemas.

All DBMS vendors also support SQL commands to describe aspects of the physical schema, but these commands are not part of the SQL language standard.

→ Information about the conceptual, external, and physical schemas is stored in the system catalogs.

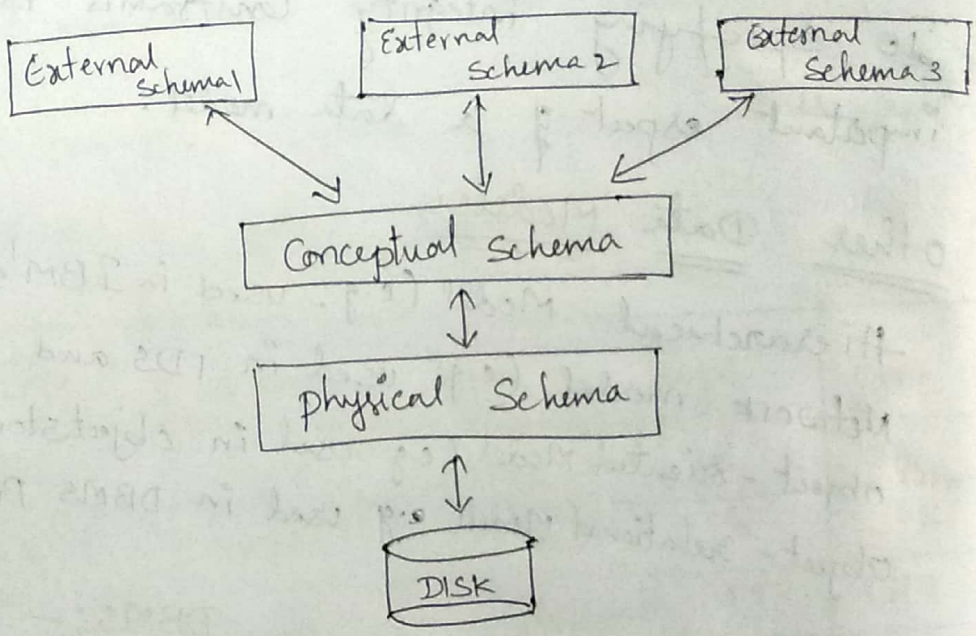

fig: Levels of Abstraction in a DBMS

Conceptual schema/ logical schema:

The conceptual schema describes the stored data in terms of the data model of the DBMS.

In a relational DBMS, the conceptual schema describes all relations that are stored in the database.

→ In our sample university database, these relations contain information about entities, such as students and faculty, and about relationships, such as student's enrollment in courses.

→ Each collection of entities and each collection of relationships can be described as a relation, leading to the following conceptual schema:

Students(sid: string, sname: string, login: string, age: integer, cgpa: real)

Faculty (fid: string, fname: string, sal: real)

Courses (cid: string, cname: string, credits: integer)

Rooms (rno: integer, address: string, capacity: integer)

Enrolled(sid: string, cid: string, grade: string)

Teaches (fid: string, cid: string)

Meets_In(cid: string, rno: integer, time: string)

The choice of relations and the choice of fields for each relation, is not always obvious, and the process of arriving at a good conceptual schema is called Conceptual database design.

Physical Schema:

The physical schema specifies additional storage details. Essentially, the physical schema summarizes how the relations described in the conceptual schema are actually stored on secondary storage devices such as disks and tapes.

We must decide what file organizations to use to store the relations and create auxiliary data structures called indexes, to speed up data retrieval operations.

→ A sample physical schema for the university database follows:

- Store all relations as unsorted files of records.
- Create indexes on the first column of the Students, Faculty, and Courses relations, the Sal column of Faculty and the Capacity column of Rooms.

→ Decisions about the physical schema are based on an understanding of how the data is typically accessed.

→ The process of arriving at a good physical schema is called physical database design.

External Schema / View Level :-

External schemas, which usually are also in terms of the data model of the DBMS, allow data access to be customized (and authorized) at the level of individual users & groups of users.

→ Any given database has exactly one conceptual schema and one physical schema because it has

just one set of stored relations, but it may have several external schemas, each tailored to a particular group of users.

→ Each external schema consists of a collection of one or more views and relations from the conceptual schema.

→ A view is conceptually a relation, but the records in a view are not stored in the DBMS.

→ The external schema design is guided by end user requirements.

Example:

We might want to allow students to find out the names of faculty members teaching courses as well as course enrollments.

This can be done by defining the following view:

Courseinfo(rid : string, fname : string, enrollment : integer)

→ A user can treat a view just like a relation and ask questions about the records in the view. Even though the records in the view are not stored explicitly, they are computed as needed.

→ We did not include Courseinfo in the conceptual schema because we can compute Courseinfo from the relations in the conceptual schema, and to store it in addition would be redundant. Such redundancy, in addition to the wasted space, could lead to inconsistencies.

3. Data Independence :-

A very important advantage of using a DBMS is that it offers data independence. That is, application programs are insulated from changes in the way the data is structured and stored.

→ Relations in the external schema are in principle generated on demand from the relations corresponding to the conceptual schema.

→ If the underlying data is reorganized, that is, the conceptual schema is changed, the definition of a view relation can be modified so that the same relation is computed as before.

Example:

Suppose the faculty relation in our university database is replaced by the following two relations:

Faculty_Public(fid: String, fname: String, office: integer)

Faculty_Private(fid: String, sal: real)

Some confidential information about faculty has been placed in a separate relation and information about offices has been added.

→ Users can be shielded from changes in the logical structure of the data, or changes in the choice of relations to be stored.
This property is called "logical data independence"

→ The conceptual schema insulates users from changes in physical storage details. This property is referred to as "physical data independence".

→ The conceptual schema hides details such as how the data is actually laid out on disk, the file structure, and the choice of indexes.

Queries:

The below Questions a user might ask:

1. what is the name of the student with student ID
2. what is the average salary of professors who teachs course ECS461
3. How many students are enrolled in ECS461
4. what fraction of students in ECS461 received a grade better than B+.
5. Is any student with a CGPA less than 3.0 enrolled in ECS461

Such questions involving the data stored in a DBMS are called Queries.

→ A DBMS provides a specialized language, called the Query language, in which queries can be posed.

→ A very attractive feature of the relational model is that it supports powerful query language.

→ "Relational calculus" is a formal query language based on mathematical logic, and queries in this language have an intuitive, precise meaning.

→ "Relational algebra" is another formal query language, based on a collection of operators for manipulating relations, which is equivalent in power to the calculus

→ A DBMS takes great care to evaluate queries as efficiently as possible.

→ A DBMS enables users to create, modify, and query data through a "data manipulation language (DML)".

Thus, the query language is only one part of the DML, which also provides constructs to insert, delete and modify data.

→ The DML and DDL are collectively referred to as the data sublanguage when embedded with in a host language (e.g: C or COBOL).

Transaction Management :-

Consider a database that holds information about airline reservations. At any given instant, it is possible that several travel agents are looking up information about available seats on various flights and making new seat reservations.

→ When several users access a database concurrently, the DBMS must order their requests carefully to avoid conflicts.

Example1:-

When one travel agent looks up Flight 100 on some given day and finds an empty seat, another travel agent may simultaneously be making a reservation for that seat, thereby making the information seen by the first agent obsolete.

Example 2:- Bank's Database

While one user's application program is computing the total deposits, another application may transfer money from an account that the first application has just 'seen' to an account that has not yet been seen, thereby causing the total to appear larger than it should be.

clearly, such anomalies should not be allowed to occur.

However, disallowing concurrent access can degrade performance.

→ The DBMS must protect users from the effects of system failures by ensuring that all data (and the status of active applications) is restored to a consistent state when the system is restarted after a crash.

→ A transaction is any one execution of a user program in a DBMS. This is the basic unit of changes as seen by the DBMS: partial transactions are not allowed, and the effect of a group of transactions is equivalent to some serial execution of all transactions.

1. Concurrent Execution of Transactions:-

An important task of a DBMS is to schedule concurrent accesses to data so that each user can safely ignore the fact that others are accessing the data concurrently.

The importance of this task cannot be underestimated because a database is typically shared by a large

number of users, who submit their requests to the DBMS independently and simply cannot be expected to deal with arbitrary changes being made concurrently by other users.

→ A DBMS allows users to think of their programs as if they were executing in isolation, one after the other in some order chosen by the DBMS

Example:

If a program that deposits cash into an account is submitted to the DBMS at the same time as another program that debits money from the same account, either of these programs could be run first by the DBMS, but their steps will not be interleaved in such a way that they interfere with each other.

→ A "Locking Protocol" is set of rules to be followed by each transaction to ensure that, even though actions of several transactions might be interleaved, the net effect is identical to executing all transactions in some serial order.

→ A 'lock' is a mechanism used to control access to database objects.

→ Two kinds of locks are commonly supported by a DBMS:

- shared locks
- exclusive lock

→ "Shared locks" on an object can be held by two different transactions at the same time.

→ "Exclusive Lock" on an object ensures that no other transactions hold any lock on this object.

2. Incomplete Transactions and System Crashes:-

→ Transactions can be interrupted before running to completion for a variety of reasons

Eg: System crash.

→ A DBMS must ensure that the changes made by such incomplete transactions are removed from the database.

→ Example:-

If the DBMS is in the middle of transferring money from account 'A' to account 'B' and has debited the first account but not yet credited the second when the crash occurs, the money debited from account A must be restored when the system comes back up after the crash.

→ The DBMS maintains a 'log' of all writes to the database.

→ A crucial property of the log is that each write action must be recorded in the log (on disk) before the corresponding change is reflected in the database itself. otherwise,

→ if the system 'crashes' just after making the change in the database but before the change is recorded in the log, the DBMS would be unable to detect and undo this change.

This property is called "Write-Ahead Log or WAL".

→ To ensure this property, the DBMS must be able to selectively force a page in memory to disk.

→ The log is also used to ensure that the changes made by a successfully completed transaction are not lost due to a system crash.

→ The time required to recover from a crash can be reduced by periodically forcing some information to disk, this periodic operation is called a "checkpoint".

→ The DBMS must ensure that the effects of all transactions that completed prior to the crash are restored, and that the effects of incomplete transactions are undone.