UNIT – I

Introduction to Database: File System Organization: Sequential, Pointer, Indexed, Direct. Purpose of Database System, Database Characteristics, Users of Database System, Advantages of DBMS Approach, Schemas and Instances, Three Schema Architecture and Data Independence, The Database System Environment, Relational Algebra

COURSE OBJECTIVES:

- To get familiar with fundamental concepts of database management such as database design, database languages, and database-system implementation

COURSE OUTCOMES:

- ✓ Develop the knowledge of fundamental concepts of database management systems.

INTRODUCTION TO DATABASE

A **database** is a structured collection of data. The data are typically organized to model relevant aspects of reality (for example, the availability of rooms in hotels), in a way that supports processes requiring this information (for example, finding a hotel with vacancies).

The term *database* is correctly applied to the data and their supporting data structures, and not to the database management system (DBMS). The database data collection with DBMS is called a database system.

The term *database system* implies that the data are managed to some level of quality (measured in terms of accuracy, availability, usability, and resilience) and this in turn often implies the use of a general-purpose database management system (DBMS).

FILE SYSTEM ORGANIZATION:

Various types of file organization include:

1. **Sequential Access:** A sequential file contains records organized by the order in which they were entered. The order of the records is fixed. Records in sequential files can be read or written only sequentially.
   - After you place a record into a sequential file, you cannot shorten, lengthen, or delete the record. However, you can update (REWRITE) a record if the length does not change. New records are added at the end of the file.
   - If the order in which you keep records in a file is not important, sequential organization is a good choice. Sequential output is also useful for printing reports.

2. **Direct Access:** Direct access file is also known as random access or relative file organization. In direct access file, all records are stored in direct access storage device, such as hard disk. The records are randomly placed throughout the file. The records are updated directly and rewritten back in the same location.
   - This file organization is useful for immediate access to large amount of information. It is used in accessing large databases. It is also called as hashing.
3. **Pointer based access:** Each file is a linked list of disk blocks. The disk blocks can be scattered anywhere on the disk. The directory entry contains a pointer to the starting and the ending file block. Each block contains a pointer to the next block occupied by the file.
   - This is very flexible in terms of file size. File size can be increased easily since the system does not have to look for a contiguous chunk of memory.
   - This method does not suffer from external fragmentation. This makes it relatively better in terms of memory utilization.
4. **Indexed file organization:** An indexed file contains records ordered by a *record key*. A record key uniquely identifies a record and determines the sequence in which it is accessed with respect to other records. Each record contains a field that contains the record key.
   - The possible record transmission (access) modes for indexed files are sequential, random, or dynamic. When indexed files are read or written sequentially, the sequence is that of the key values.
   - An indexed file can also use *alternate indexes*, for example, you could access a file through employee department rather than through employee number.

PURPOSE OF DATABASE SYSTEM

The typical file processing system is supported by a conventional operating system. The system stores permanent records in various files, and it needs different application programs to extract records from, and add records to, the appropriate files.

A file processing system has a number of major disadvantages.

1. Data Redundancy and Inconsistency: In file processing, every user group maintains its own files for handling its data processing applications. For example, consider the UNIVERSITY database. Here, two groups of users might be the course registration personnel and the accounting office. The accounting office also keeps data on registration and related billing information, whereas the registration office keeps track of student courses and grades. Storing the same data multiple times is called data redundancy. This redundancy leads to several problems:

- ✓ Need to perform a single logical update multiple times.
- ✓ Storage space is wasted.
- ✓ Files that represent the same data may become inconsistent.
- ✓ Data inconsistency in the various copies of the same data may no longer agree.

Example: One user group may enter a student's birth date erroneously as JAN-24-1995, whereas the other user groups may enter the correct value of JAN-29-1995.

2. Difficulty in Accessing Data: File processing environments do not allow needed data to be retrieved in a convenient and efficient manner. Suppose that one of the bank officers needs to find out the names of all customers who live within a particular area. The bank officer has now two choices: either obtain the list of all customers and extract the needed information manually or ask a system programmer to write the necessary application program. Both alternatives are obviously unsatisfactory. Suppose that such a program is written, and that, several days later, the same officer needs to trim that list to include only those customers who have an account balance of $10,000 or more. A program to generate such a list does not exist. Again, the officer has the preceding two options, neither of which is satisfactory.

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

4. Integrity Problems: The data values stored in the database must satisfy certain types of consistency constraints. Example: The balance of certain types of bank accounts may never fall below a prescribed amount. Developers enforce these constraints in the system by addition appropriate code in the various application programs

5. Atomicity Problems: Atomic means the transaction must happen in its entirety or not at all. It is difficult to ensure atomicity in a conventional file processing system. Example: Consider a program to transfer $50 from account A to account B. If a system failure occurs during the execution of the program, it is possible that the $50 was removed from account A but was not credited to account B, resulting in an inconsistent database state.

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. In such an environment, interaction of concurrent updates is possible and may result in inconsistent data. To guard against this possibility, the system must maintain some form of supervision. But supervision is difficult to provide because data may be accessed by many different application programs that have not been coordinated previously. Example: When several reservation clerks try to assign a seat on an airline flight, the system should ensure that each seat can be accessed by only one clerk at a time for assignment to a passenger.

7. Security Problems: Enforcing security constraints to the file processing system is difficult.

DATABASE SYSTEM APPLICATIONS

- ✓ Banking: All Transactions
- ✓ Universities: Registration, Grades
- ✓ Sales: Customers, Products, Purchases
- ✓ Online Retailers: Order Tracking, Customized Recommendations
- ✓ Manufacturing: Production, Inventory, Orders, Supply Chain
- ✓ Airlines: Reservations, Schedules
- ✓ Human Resources: Employee Records, Salaries, Tax Deductions

DATABASE CHARACTERISTICS

1. **Real-world entity:** Data management systems have been designed keeping in mind the needs of business organizations. They help businesses manage large amounts of data efficiently. These systems store huge volumes of data and provide easy ways to search through them. Some examples of such applications are Microsoft Access, Oracle, MySQL, etc.
2. **Relational databases:** Relational databases were first introduced in the 1970s. In this type of database, each record contains fields called attributes. Each attribute represents one piece of information about a particular object.
    - For example, if you want to keep track of your personal details then you will need three different attributes namely name, address, phone number. All of these attributes together form a single row in a table. This means that every time we add new information into our database, we must insert multiple rows into the same table.
    - If we do not follow this rule then we may end up having duplicate entries in our database. So relational databases allow us to organize data using relations between objects.
3. **Structured query language:** Structured Query Language was developed in the 1980s. It provides a way to write queries against a database. Queries written in SQL are known as structured queries because they use predefined structures to define relationships among entities.

    An example would be: SELECT * FROM employees WHERE department = 'IT'.

    Here, the word '*' stands for all columns from the employee table.

    We can also specify only certain columns while selecting other ones.
4. **Isolation of data and application:** A database system is not the same thing as its data. A database works and organizes, whereas data is said to be passive. Metadata, which is data about data, is stored by the database management system.

    Also, in a traditional file management system, the structure of data files is defined in the application programs so the user had to change everything. But in DBMS, the structure of data files is not stored in the program but it is stored in the system catalog. Internal improvement of data efficiency or any changes in the data doesn't have an effect on application software with the help of this- that is, it offers insulation between Data and Program.
5. **ACID properties:** DBMS adheres to the concepts of Atomicity, Consistency, Isolation, and Durability, or ACID Properties. These concepts are applied to transactions, which operate and play around with data in a database. In multi-transactional environments, ACID properties help the database stay healthy in case of failure.
6. **Multiuser and concurrent access:** Data can be accessed and manipulated in parallel with the help of the DBMS. Users are unaware of the restrictions on transactions when handling the same data item.

    DBMS offers multiple views for various clients. A client who is in the Sales division will have an alternate perspective on the database than an individual working in the Production office. This component empowers the clients to have a concentrated perspective on the database as indicated by their prerequisites.

7. **Object oriented programming:** Object oriented programming is another technique used to design programs. Objects contain properties and methods. Properties represent variables or constants which hold values. Methods are procedures that operate on those properties. The main advantage of OOP over procedural programming is that it allows developers to create reusable components by encapsulating their functionality within classes. Classes are collections of related code and data elements. Developers can reuse existing classes instead of writing similar pieces of code again and again.
8. **Transactional processing:** Transactions are an essential part of any application. Transactions ensure consistency across multiple users accessing the system at the same time. When transactions fail due to errors, the entire transaction should roll back so that no changes made during the failed operation remain permanent. Transaction processing ensures that when there is a problem, everything remains consistent.
9. **Data mining:** Data mining refers to techniques used to analyze vast quantities of unstructured data. There are many types of data mining algorithms like classification, clustering, association rules, regression analysis, decision trees, neural networks, genetic algorithms, etc.
10. **Distributed database systems:** Distributed databases store information across several computers connected through a computer network. This type of architecture makes it easy to add more servers without having to rebuild the whole software infrastructure. A distributed database has three parts: client applications, server applications, and shared storage. Client applications access the shared storage using standard protocols such as TCP/IP. Server applications provide services to clients. Shared storage provides persistent storage space where all the data resides. In addition, each node stores metadata about other nodes in the cluster.
11. **Less redundancy and More consistency:** The DBMS follows the rules of normalization, which splits a relation when there is more than one attribute with the same value. Normalization is a method of reducing data redundancy. Every relation in a database is consistent, that's the state of consistency. Attempts to leave databases in different states can be detected with methods and techniques. A database management system can give greater consistency than earlier forms of data storage.
12. **Data security and integrity:** DBMS gives security to the information put away in this is on the grounds that all clients have different rights to access the database. A portion of the client can get to the entire information in the database while others can get to a small segment of the database.
    - One of the main attributes of a DBMS - Integrity guarantees the quality and dependability of the database framework. It protects from unapproved access to the database and makes it safer. It allows only consistent and exact information into the database.

USERS OF DATABASE SYSTEM

Primary goal of a database system is to retrieve information from and store new information into the database. People who work with a database can be categorized as database users or database administrators.

Database Users: There are four different types of database-system users, differentiated by the way they expect to interact with the system.

1) Naïve Users: Naïve users are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously. The typical user interface for naïve users is a forms interface, where the user can fill in appropriate fields of the form. Naïve users may also simply read reports generated from the database.

2) Application Programmers: Application programmers are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces. Rapid application development (RAD) tools are tools that enable an application programmer to construct forms and reports with minimal programming effort.

3) Sophisticated Users: Sophisticated users interact with the system without writing programs. Instead, they form their requests either using a database query language or by using tools such as data analysis software. Analysts who submit queries to explore data in the database fall in this category.

4) Specialized Users: Specialized users are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework. Among these applications are computer-aided design systems, systems that store data with complex data types (for example, graphics data and audio data), and environment-modeling systems.

Database Administrator:

One of the main reasons for using DBMS is to have central control of both the data and the programs that access those data. ***A person who has such central control over the system is called a database administrator (DBA)***. The **functions/responsibilities** of a DBA include:

1. **Schema Definition:** The DBA creates the original database schema by executing a set of data definition statements in the DDL.
2. **Storage Structure and Access-Method Definition**
3. **Schema and Physical-Organization Modification:** The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.
4. **Granting of Authorization for Data Access:** By granting different types of authorization, the database administrator can regulate which parts of the database various users can access. The authorization information is kept in a special system structure that the database system consults whenever someone attempts to access the data in the system.
5. **Routine Maintenance:** Examples of the database administrator's routine maintenance activities are:
   - ✓ Periodically backing up the database, to prevent loss of data in case of disasters such as flooding.
   - ✓ Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required.
   - ✓ Monitoring jobs running on the database and ensuring that performance is not degraded by very expensive tasks submitted by some users.

ADVANTAGES OF DBMS APPROACH

Various advantages of DBMS approach are:

1. **Better Data Transferring:** Database management creates a place where users have an advantage of more and better managed data. Thus making it possible for end-users to have a quick look and to respond fast to any changes made in their environment.
2. **Better Data Security:** As number of users increases data transferring or data sharing rate also increases thus increasing the risk of data security. It is widely used in corporate world where companies invest money, time and effort in large amount to ensure data is secure and is used properly. DBMS provide a better platform for data privacy and security policies thus, helping companies to improve Data Security.
3. **Better data integration:** Due to Database Management System we have an access to well managed and synchronized form of data thus it makes data handling very easy and gives integrated view of how a particular organization is working and also helps to keep a track on how one segment of the company affects other segment.
4. **Minimized Data Inconsistency:** Data inconsistency occurs between files when different versions of the same data appear in different places. For Example, data inconsistency occurs when a student name is saved as "John Wayne" on a main computer of school but on teacher registered system same student name is "William J. Wayne", or when the price of a product is $86.95 in local system of company and its National sales office system shows the same product price as $84.95. So if a database is properly designed then Data inconsistency can be greatly reduced hence minimizing data inconsistency.
5. **Faster data Access:** The Data base management system (DBMS) helps to produce quick answers to database queries thus making data accessing faster and more accurate. For example, to read or update the data. For example, end users, when dealing with large amounts of sale data, will have enhanced access to the data, enabling faster sales cycle.

   Some queries may be like:

   - What is the increase of the sale in last three months?
   - What is the bonus given to each of the salespeople in last five months?

6. **Better decision making:** Due to DBMS now we have better managed data and improved data accessing because of which we can generate better quality information hence on this basis better decisions can be made. Better Data quality improves accuracy, validity and time it takes to read data. DBMS does not guarantee data quality, it provides a framework to make it is easy to improve data quality.
7. **Increased end-user productivity:** The data which is available with the help of combination of tools which transform data into useful information helps end user to make quick, informative and better decisions that can make difference between success and failure in the global economy.
8. **Simple:** Data base management system (DBMS) gives simple and clear logical view of data

SCHEMAS AND INSTANCES

Schema: The logical structure of the database (or) the overall design of the database is called the database schema. Schemas are changed infrequently, if at all. A database schema corresponds to the variable declarations (along with associated type definitions) in a program. Types of Schema include:

Physical Schema: describes the database design at the physical level

Logical Schema: describes the database design at the logical level.

A database may also have several schemas at the view level, sometimes called ***subschemas*** that describe different views of the database.

- ✓ The logical schema is the most important, in terms of its effect on application programs, since programmers construct applications by using the logical schema.

Instance: Collection of information stored in the database at a particular moment is called an instance of the database. Values of variables in a program at a point in time relate to an instance of database schema.

Physical Data Independence: The ability to modify the physical schema without changing the logical schema is known as Physical data independence. Usually, applications depend on the logical schema.

Logical Data Independence: Ability to modify logical schema without changing physical schema. It is harder to achieve as application programs are usually heavily dependent on logical structure of data.

THREE SCHEMA ARCHITECTURE

Data Abstraction: A major purpose of a database system is to provide users with an abstract view of the data. That is, the system hides certain details of how the data are stored and maintained. For the system to be usable, it must retrieve data efficiently. There are several levels of abstraction:

1) **Physical Level:** The lowest level of abstraction describes ***how*** the data are actually stored. The physical level describes complex low-level data structures in detail.
2) **Logical Level:** The next-higher level of abstraction describes ***what*** data are stored in the database, and what relationships exist among those data. The logical level thus describes the entire database in terms of a small number of relatively simple structures. Database administrators, who must decide what information to keep in the database, use the logical level of abstraction.
3) **View Level:** Highest level of abstraction describes part of entire database for a particular group of users. Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database. Many users of the database system do not need all this information; instead, they need to access only a part of the database. View level of abstraction exists to simplify their interaction with the system. ***The system may provide many views for same database.***

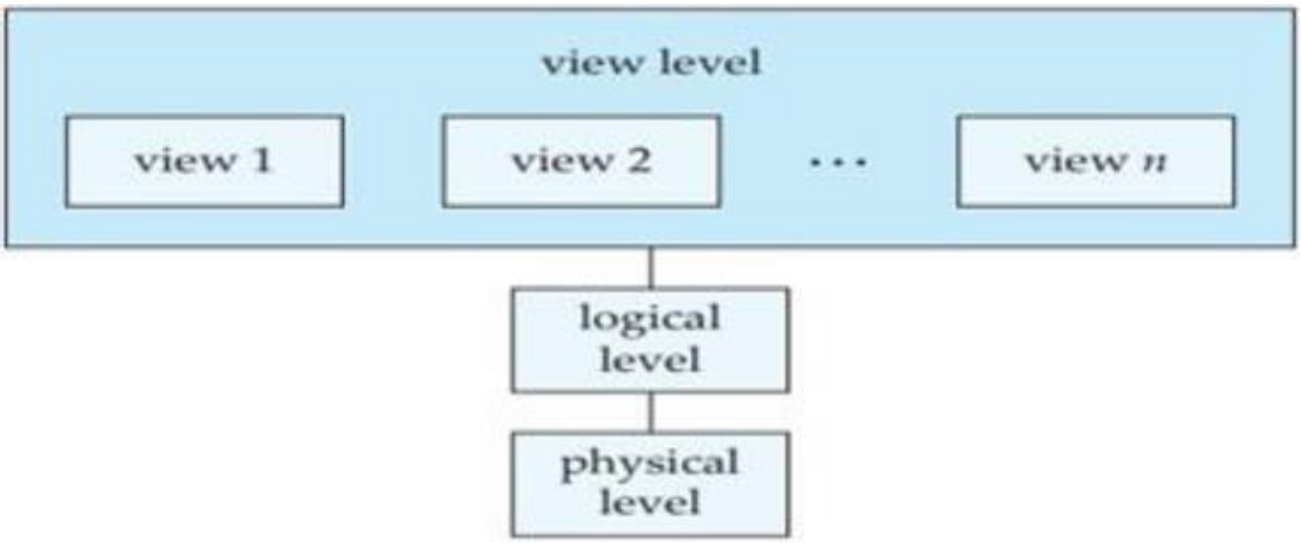

Fig: Three Levels of Data Abstraction

DATABASE SYSTEM ENVIRONMENT

One of the primary aims of a database is to supply users with an abstract view of data, hiding a certain element of how data is stored and manipulated. Therefore, the starting point for the design of a database should be an abstract and general description of the information needs of the organization that is to be represented in the database. And hence you will require an environment to store data and make it work as a database.

Storage Manager: Storage manager is the component of a database system that provides interface between low-level data stored in database, application programs and queries submitted to the system.

The storage manager is important because databases typically require a large amount of storage space. Corporate databases range in size from hundreds of gigabytes to, for the largest databases, terabytes of data. Since the main memory of computers cannot store this much information, the information is stored on disks. Data are moved between disk storage and main memory as needed.

- *The storage manager is responsible for the interaction with the file manager*. The raw data are stored on the disk using the file system provided by the operating system. The storage manager is responsible for storing, retrieving, and updating data in the database.

The storage manager components include:

1) **Authorization and Integrity Manager**, which tests for the satisfaction of integrity constraints and checks the authority of users to access data.
2) **Transaction Manager**, which ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.
3) **File Manager**, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

4) **Buffer Manager**, which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory. Buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory.

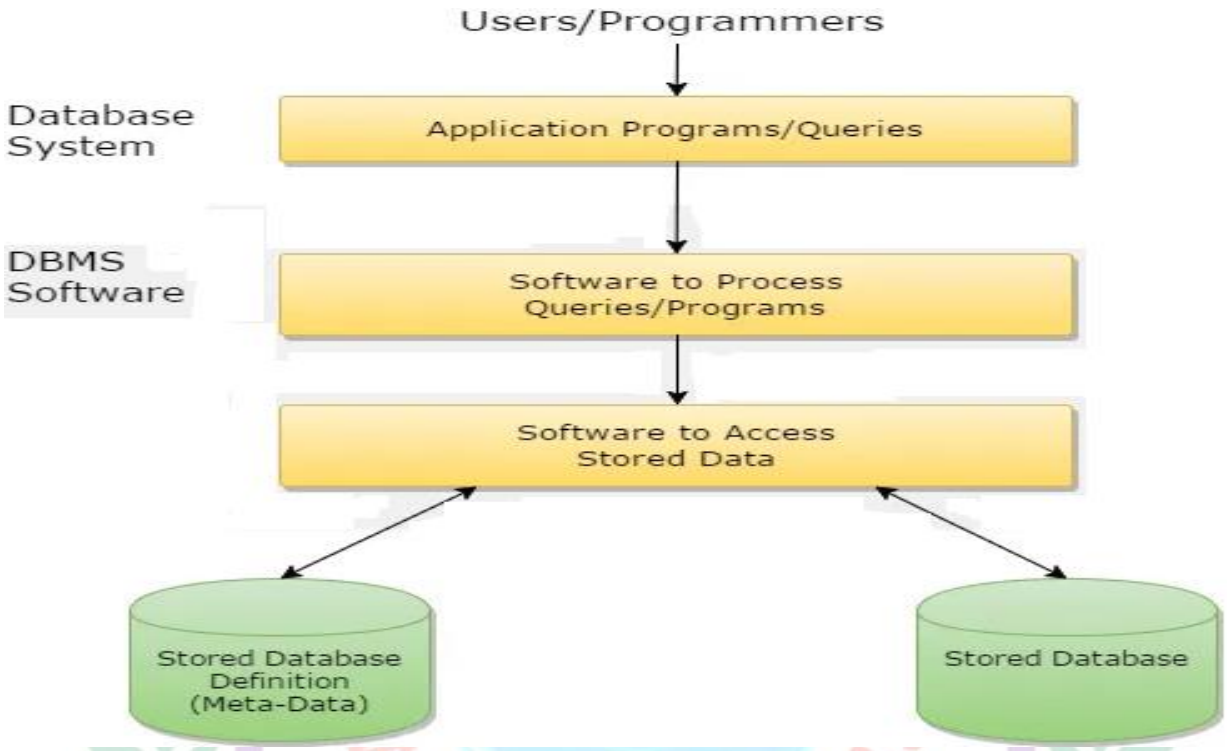

Fig: DataBase System Environment

The storage manager implements ***several data structures*** as part of the physical system implementation:

- Data files, which store the database itself.
- Data dictionary, which stores metadata about structure of database, in particular schema of database.
- Indices, which can provide fast access to data items. Like the index in a textbook, a database index provides pointers to those data items that hold a particular value.

The Query Processor: The query processor is important because it helps the database system to simplify and facilitate access to data. It allows database users to obtain good performance while being able to work at the view level. It is the job of the database system to translate updates and queries written in a nonprocedural language, at the logical level, into an efficient sequence of operations at the physical level. The query processor components include:

1) **DDL interpreter**, which interprets DDL statements and records the definitions in the data dictionary.
2) **DML compiler**, which translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands. The DML compiler also performs *query optimization*; that is, it picks lowest cost evaluation plan among the alternatives.
3) **Query evaluation engine**, which executes low-level instructions generated by the DML compiler.

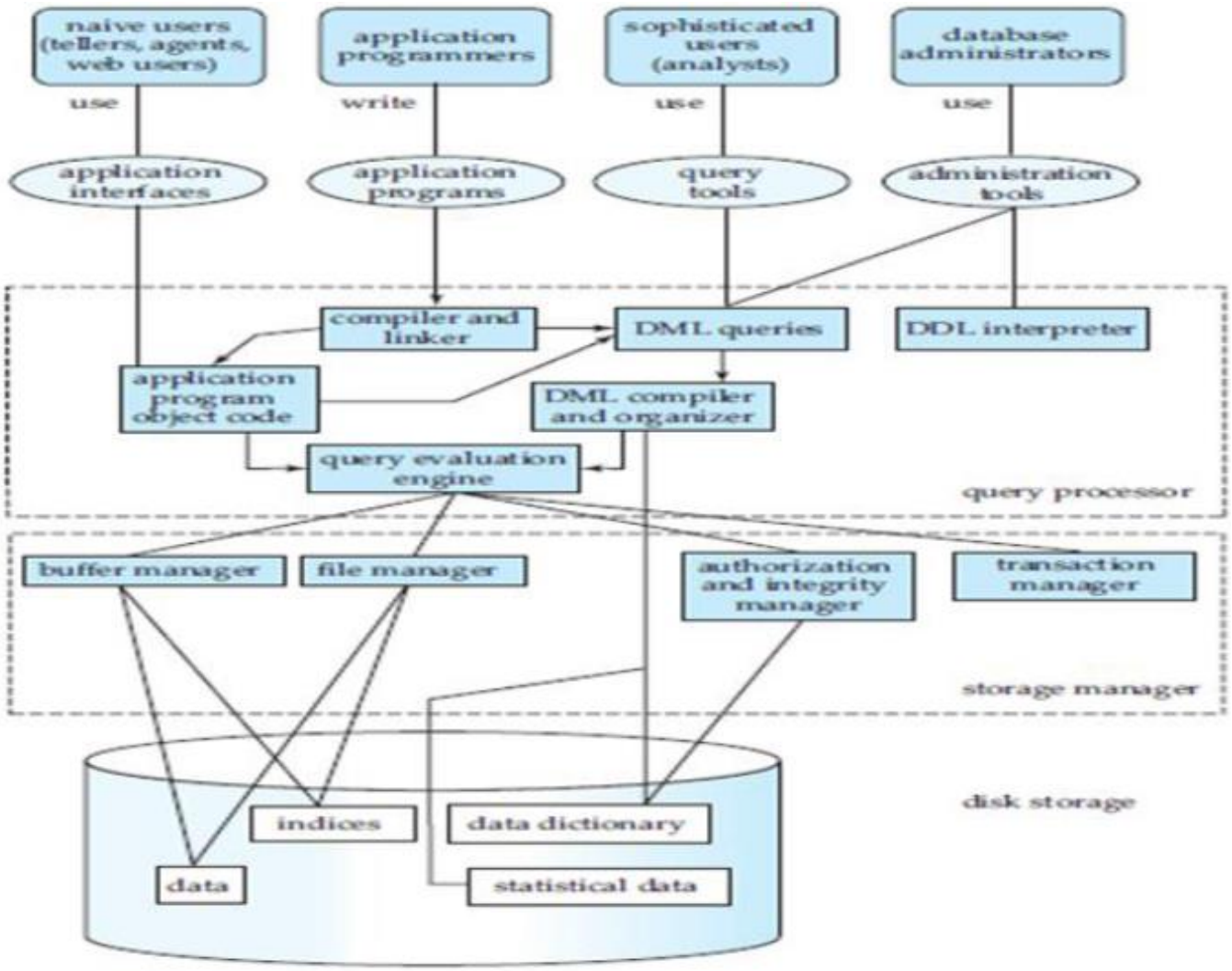

Fig: Database System Structure

RELATIONAL ALGEBRA

The relational algebra is a *procedural* query language. It consists of a set of operations that take one or two relations as input and produce a new relation as their result. The fundamental operations in the relational algebra are *select*, *project*, *union*, *set difference*, *Cartesian product*, and *rename*. In addition to the fundamental operations, there are several other operations—namely, *set intersection*, *natural join*, and *assignment*. We shall define these operations in terms of the fundamental operations.

Fundamental Operations

The select, project, and rename operations are called *unary* operations, because they operate on one relation. The other three operations operate on pairs of relations and are, therefore, called *binary* operations.

Consider the Two relations STUDENT, QUARTERLY. The STUDENT relation is used to describe the complete personal information about student, his roll no, name, date of birth, 2nd language. Another relation QUARTERLY used to describe the students marks in 3 subjects with roll no's.

STUDENT

Roll-No	Name	Date-of-Birth	Second-Language
1	Sunny	01-07-70	Hindi
2	Rashni	15-08-72	Sanskrit
3	Anthra	29-01-71	Hindi
4	Nasreen	31-12-70	Telugu

QUARTERLY

Roll-No	Maths	Physics	Computers
1	72	85	90
2	65	74	68
3	97	94	96
4	87	93	72

The following are queries based on relational algebra to obtain required information fromstored relationaldatabase.

1) **The SELECT Operation:** The Select operation selects tuples that satisfy a given predicate. Alower case Greek letter sigma (σ) is used to denote Select operation. Predicate appears as subscript tor.

The argument relation is given in parentheses. The General form of selection operation is:

$\sigma_{\text{predicate}}$ (relation)

All comparisons =, #, <, >, <=, >= were allowed in the select operation predicate. Furthermore, several predicates may be combined into a large predicate using the connectives *and* (∧) and *or* (V).

Ex: 1) List out the complete information about all students whose 2nd language is Hindi

$\sigma_{\text{2nd-language = Hindi}}$ (STUDENT)

Result of the above Query is:

Roll-No	Name	Date-of-Birth	Second-Language
1	Sunny	01-07-70	Hindi
3	Anthra	29-01-71	Hindi

Ex: 2) Display all students with Roll no with their marks who secured more than 90 in all the three subjects

σ ((Maths > 90) ∧(Physics > 90) ∧(Computers > 90))

(QUARTERLY) Result of the above Query is:

Roll-No	Maths	Physics	Computers
3	97	94	96

2) **The PROJECT Operation:** The projection of a relation is defined as a projection of all its tuples over some set of attributes. i.e., it yields a "vertical subset" of the relation. The projection operation is used to either reduce the number of attributes in the resultant relation or to reorder attributes. Projection is denoted by Greek letter pi (Π). We list these attributes that we wish to appear in the result as a subscript to. The argument

relation follows inparenthesis.

General form of projection operation is

$$\Pi_{\text{List-of-attributes}}(\text{Predicate})(\text{relation})$$

Ex: 1) List out all Roll Nos. and their ComputerMarks

$\Pi_{\text{Roll No, Computers}}$ (QUARTERLY)

Result of the above Query is:

Roll-No	Computers
1	90
2	68
3	96
4	72

2) Display all the student names with their date of birth whose 2nd language isHindi.

$\Pi_{\text{Name, Date-of-birth}}(\sigma_{\text{2nd-language = Hindi}})$ (STUDENT) Result of the above query is:

Name	Date-of-Birth
Sunny	01-07-70
Anthra	29-01-71

3) What is the Date of Birth ofRashni?

$\Pi_{\text{Date-of-Birth}}(\sigma_{\text{Name = Rashni}})$ (STUDENT)

Result of the above query is:

Date-of-Birth
15-08-72

4) Find all Roll Nos who secured more than 90 marks inComputers

$\Pi_{\text{Roll-No}}(\sigma_{\text{computers > 90}})$ (QUARTERLY)

Result of the above query is:

Roll-No
1
3

3) The RENAME Operation: Unlike relations in the DB, the results of relational-algebra expressions do not have a name that we can use to refer to them. It is useful to be able to give them names; the **rename** operator, denoted by the lowercase Greek letter rho (ρ), lets us do this. Given a relational- algebra expression *E*, the following expression returns the result of expression *E* under the name*x*.

$\rho_x(E)$

Ex: ρ_{teacher} (*instructor*)

A relation *r* by itself is considered a (trivial) relational-algebra expression. Thus, we can

also apply the rename operation to a relation *r* to get the same relation under a new name.

A Second form of the rename operation is as follows: Assume that a relational algebra expression *E* has arity *n*. Then, the following expression returns the result of expression *E* under the name *x*, and with the attributes renamed to *A*1, *A*2, . . . , *An*.

$$\rho_{x(A1,A2,\ldots,An)}(E)$$

Ex: $\rho_{\text{teacher (id, name, sal)}}(instructor)$

4) **CARTESIAN PRODUCT Operation:** This operation allows us to combine information from several relations. Thus operation is denoted by a cross(X). Thus operation is a binary. Suppose r1 and r2 are two relations, Cartesian product of these two relations can be written are r1 Xr2.

In other words, Cartesian product of two relations is the concatenation of tuples belonging to the two relations. A new resultant relation scheme is created consisting of all possible combinations of tuples.

If there are m tuples in relation r1, and n tuples in relation r2, then there is m x n ways of choosing a pair of tuples. One tuple from each relation is chosen, so there are n1 x n2 tuples in r.

Ex: (i) Find student names and their Computer marks.

To list out the Name, Computer Marks, we have to refer both the relations, STUDENT & QUARTERLY. Student name is an attribute from STUDENT relation and Computers is an attribute from QUARTERLY relation. Referring 2 relations is denoted by"X"

STUDENT X QUARTERLY

Roll-No	Name	Date-of-Birth	Second-Language	Roll-No	Maths	Physics	Computers
1	Sunny	01-07-70	Hindi	1	72	85	90
1	Sunny	01-07-70	Hindi	2	65	74	68
1	Sunny	01-07-70	Hindi	3	97	94	96
1	Sunny	01-07-70	Hindi	4	87	93	72
2	Rashni	15-08-72	Sanskrit	1	72	85	90
2	Rashni	15-08-72	Sanskrit	2	65	74	68
2	Rashni	15-08-72	Sanskrit	3	97	94	96
2	Rashni	15-08-72	Sanskrit	4	87	93	72
3	Anthra	29-01-71	Hindi	1	72	85	90
3	Anthra	29-01-71	Hindi	2	65	74	68
3	Anthra	29-01-71	Hindi	3	97	94	96
3	Anthra	29-01-71	Hindi	4	87	93	72
4	Nasreen	31-12-70	Telugu	1	72	85	90

4	Nasreen	31-12-70	Telugu	2	65	74	68
4	Nasreen	31-12-70	Telugu	3	97	94	96
4	Nasreen	31-12-70	Telugu	4	87	93	72

Information is retrieved from the above relation STUDENT X QUARTERLY. By selecting the common attribute in the same relation i.e., in given two relations, Roll No is the common attribute.

STUDENT.ROLLNO = QUARTERLY.ROLLNO

Π Name, Computers (σ STUDENT.ROLL-NO = QUARTERLY.ROLL-NO) (STUDENT X QUARTERLY)

Name	Computers
Sunny	90
Rashni	68
Anthra	96
Nasreen	72

(ii) Find the Student Roll No, Date of Birth, 2nd Language, Maths, Physics and Computer Marks.

σ STUDENT.ROLL NO = QUARTERLY.ROLL NO. (STUDENT X QUARTERLY)

Result of the query is:

Roll-No	Name	Date-of-Birth	Second-Language	Roll-No	Maths	Physics	Computers
1	Sunny	01-07-70	Hindi	1	72	85	90
2	Rashni	15-08-72	Sanskrit	2	65	74	68
3	Anthra	29-01-71	Hindi	3	97	94	96
4	Nasreen	31-12-70	Telugu	4	87	93	72

5) **UNIONOperation:**Theunionoftworelationsrandsisdenotedbyr∪s.TheoutputrelationZ=r ∪s has tuples drawn from r and s. The result relation Z contains tuples that are in either r or s or in both of them. The duplicate tuples are eliminated.

For a union operation $r \cup s$ to be valid, we require that two conditions hold:

- The relations *r* and *s* must be of the same arity. i.e., they must have the same number ofattributes.
- The domains of the *i*th attribute of *r* and the *i*th attribute of *s* must be the same, for all*i*.

Note that *r* and *s* can be either database relations or temporary relations that are the result of relational algebra expressions.

As an example, consider the relations CULTURAL (name, class) and SPORTS (name,

class). These two relations represent information about all cultural competition winners & sports winnersseparately.

CULTURAL

Name	Class
Kavya	MPC III
Lahari	MSC II
Bhanu	MCA II
Zeba	MPC III
Nisha	MBA II
Hima	MPC III

SPORTS

Name	Class
Deepta	MCA III
Lahari	MSC II
Hima	MPC III
Archana	MBA II
Sheela	MPC III

Ex: Find all the student Names of MPC III who won cultural competition or sports competition or both competitions.

$\Pi_{\text{Name}}(\sigma_{\text{Class = 'MPC III'}})$ (CULTURAL) $\cup \Pi_{\text{Name}}(\sigma_{\text{Class = MPC III}})$ (SPORTS)

NAME
Kavya
Zeba
Hima

NAME
Hima
Sheela

NAME
Kavya
Zeba
Hima
Sheela

6) SET-DIFFERENCE Operation: The difference between two relations r and s is r - s. The result relation contains the set of tuples belonging to r and not ins.

Ex: Find Student Names of MPC III who won cultural prizes but not sports.

$\Pi_{\text{Name}}(\sigma_{\text{Class = 'MPC III'}})$ (CULTURAL) - $\Pi_{\text{Name}}(\sigma_{\text{Class = MPC III}})$ (SPORTS)

NAME
Kavya
Zeba
Hima

NAME
Hima
Sheela

NAME
Kavya
Zeba

Formal Definition of the Relational Algebra: The fundamental operations of relational algebra allow us to give a complete definition of an expression in the relational algebra. A basic expression in the relational algebra consists of either one of the following:

- A relation in thedatabase
- A constantrelation

A constant relation is written by listing its tuples within { }, for example

{(22222, Einstein, Physics, 95000), (76543, Singh, Finance, 80000)}.

A general expression in the relational algebra is constructed out of smaller sub expressions. Let *E*1 and

*E*2 be relational-algebra expressions. Then, the following are all relational-algebra expressions:

- ✓ $E1 \cup E2$
- ✓ $E1 - E2$
- ✓ $E1 \times E2$
- ✓ $\sigma_P(E1)$, where P is a predicate on attributes in $E1$
- ✓ $\Pi_S(E1)$, where S is a list consisting of some of the attributes in $E1$
- ✓ $\rho_x(E1)$, where x is the new name for the result of $E1$

Additional Relational-Algebra Operations

We define additional operations that do not add any power to the algebra, but simplify common queries.

1) **SET-INTERSECTION Operation:** The intersection of two relations r and s is denoted by r ∩s. The output relation contains the set of all tuples belonging to both r ands.

Ex: List out all the names belonging to MPC III who won both the cultural & sports competition.

$$\Pi_{\text{Name}}(\sigma_{\text{Class = 'MPC III'}})(\text{CULTURAL}) \cap \Pi_{\text{Name}}(\sigma_{\text{Class = MPC III}})(\text{SPORTS})$$

NAME
Kavya
Zeba
Hima

NAME
Hima
Sheela

NAME
Hima

2) **NATURAL JOIN Operation:** Natural join is a binary operation that allows us to combine certain selections and a Cartesian product into one operation. It is denoted by the "join" symbol. Natural join operation forms a Cartesian product of its two arguments, performs a selection forcing equalities on those attributes that appear in both relation schemas, and finally removes duplicatecolumns.

Ex:STUDENT ⋈ QUARTERLY

This operation performs a Cartesian product (X) of two relations, performs selection equality on those attributes that appear in both the relation schemas and finally removes duplicate columns. In STUDENT X QUARTERLY relations ROLLNO is common in both relations.

i.e.STUDENT ⋈ QUARTERLY becomes:

Roll-No	Name	Date-of-Birth	Second-Language	Roll-No	Maths	Physics	Computers
1	Sunny	01-07-70	Hindi	1	72	85	90
2	Rashni	15-08-72	Sanskrit	2	65	74	68

3	Anthra	29-01-71	Hindi	3	97	94	96
4	Nasreen	31-12-70	Telugu	4	87	93	72

Query can be written as

$\Pi_{\text{Roll No, Name, Date of Birth, 2nd language, Maths, Physics, Computers}}$ (STUDENT $\bowtie$ QUARTERLY) Now, to find student names and Computer marks, the query willbe:

$\Pi_{\text{Name,Computers}}$(STUDENT QUARTERLY) $\bowtie$

Name	Computers
Sunny	90
Rashni	68
Anthra	96
Nasreen	72

3) **The ASSIGNMENT Operation:** It is convenient at times to write a relational-algebra expression by assigning parts of it to temporary relation variables. The **assignment** operation, denoted by ←, works like assignment in a programming language. To illustrate this operation, consider the definition of the natural-join operation. We could write $r \cup s$as:

$$result \leftarrow r \cup s$$

The evaluation of an assignment does not result in any relation being displayed to the user. Rather, the result of the expression to the right of the ← is assigned to the relation variable on the left of the ←.

This relation variable may be used in subsequent expressions.

For relational algebra queries, assignment must always be made to a temporary relation variable. Note that the assignment operation does not provide any additional power to the algebra. It is, however, a convenient way to express complex queries.

Extended Relational-Algebra Operations

The relational algebra operations that provide the ability to write queries that cannot be expressed using the basic relational-algebra operations are called **extended relational-algebra** operations.

1) **GENERALIZED PROJECTION:** The first operation is the **generalized-projection** operation, which extends the projection operation by allowing operations such as arithmetic and string functions to be used in the projection list. The generalized-projection operation has theform:

$$\Pi_{F1,F2,\ldots,Fn}(E)$$

where E is any relational-algebra expression, and each of $F1, F2, \ldots, Fn$ is an arithmetic expression involving constants and attributes in the schema of E. As a base case, the expression may be simply an attribute or a constant. In general, an expression can use arithmetic operations such as $+, -, *$, and $\div$ on

numeric valued attributes, numeric constants, and on expressions that generate a numeric result. Generalized projection also permits operations on other data types, such as concatenation of strings.
For example, the expression:

$$\Pi_{ID,\ name,\ deptname,\ salary \div 12}(instructor)$$

gives the *ID*, *name*, *deptname*, and the monthly salary of each instructor.

2) **AGGREGATION:** The second extended relational-algebra operation is the aggregate operation *G*, which permits the use of aggregate functions such as min or average, on sets ofvalues.

Aggregate Functions: Aggregate Functions take a collection of values and return a single value as a result. Aggregate operation in relational algebra is expressed as:

$G_1, G_2, \ldots, Gn\ G_{F1(A1), F2(A2),\ldots, Fn(An)}(E)$

Where, *E* is any relational algebra expression Each $F_i$is an aggregate function
Each $A_i$is an attribute name

$G_1, G_2 \ldots, G_n$ is a list of attributes on which to group (can be empty) The Aggregate Functions include:

1. **sum:** It is used to find the sum of values of an attribute in a relation. Ex: $G_{\mathbf{sum}(salary)}(instructor)$
2. **avg**: It is used to find the average value of an attribute in a relation. Ex: $G_{\mathbf{avg}(salary)}(instructor)$
3. **count**: It is used to find the number of values in an attribute in a relation. Ex: $G_{\mathbf{count}(salary)}(instructor)$

There are cases where we must eliminate multiple occurrences of a value before computing an aggregate function. If we do want to eliminate duplicates, we use the same function names as before, with the addition of the key word "**distinct**" appended to the end of the function name (for ex. **count-distinct**). Now, the above example can also be written as:

$G_{\mathbf{count\text{-}distnict}(salary)}(instructor)$

4. **min:** It is used to find the minimum value in an attribute in arelation. Ex: $G_{\mathbf{min}(salary)}(instructor)$
5. **max:** It is used to find the maximum value in an attribute in arelation. Ex: $G_{\mathbf{max}(salary)}(instructor)$

❖ Result of aggregation does not have aname.
- Can use rename operation to give it a name.
- For convenience, we permit renaming as part of aggregate operation

NULL VALUES: It is possible for tuples to have a null value, denoted by *null*, for some of their attributes. *null* signifies an unknown value or that a value does not exist.

✓ The result of any arithmetic expression involving *null* is*null.*

- ✓ Aggregate functions simply ignore null values
    - Is an arbitrary decision. Could have returned null as result instead.
    - We follow the semantics of SQL in its handling of null values.
- ✓ For duplicate elimination and grouping, null is treated like any other value, and two nulls are assumed to be the same. As an alternative, assume each null is different from each other like inSQL.

Arithmetic Operations and Comparisons with null values return the special truth value *unknown or null.*

For logical operators with an input as null:

OR: (*unknown* **or** *true*) = *true,*
(*unknown* **or** *false*) = *unknown,*
(*unknown* **or** *unknown) = unknown*

AND: *(true* **and** *unknown) = unknown,*
(false **and** *unknown) = false,*
(unknown **and** *unknown) = unknown*

NOT: *(***not** *unknown) = unknown*