DATA is a collection of facts and figures that can be processed to produce information.

Database is a collection of related data.

a database is a structured collection of data that is organized and stored in a way that allows for efficient retrieval, updating, and management of information. A database typically consists of one or more tables, which are made up of rows and columns.

A database is a collection of related data. By data, we mean known facts that can be recorded and that have implicit meaning.

There are various types of databases, but the two most common models are:

1. Relational Databases: These databases store data in tabular form, and relationships between the tables are defined through keys. The Structured Query Language (SQL) is used to interact with relational databases
2. NoSQL Databases: Unlike relational databases, NoSQL databases use different data models and structures, making them more flexible for handling unstructured or semi-structured data.

Databases play a vital role in various applications and industries, ranging from web applications, mobile apps, enterprise systems, and scientific research to financial systems, healthcare, and more. They offer a reliable and organized way to store, manage, and retrieve data, ensuring data consistency, integrity, and security.

Database management refers to the process of efficiently and securely storing, organizing, maintaining, and retrieving data within a database system. . The primary goal of database management is to provide a reliable, scalable, and secure platform for applications and users to interact with data effectively.

A database management system (DBMS) is a collection of programs that enables users to create and maintain a database. It is a software application that enables users and applications to interact with a database. The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications.

Defining a database involves specifying the data types, structures, and constraints of the data to be stored in the database.

The database definition or descriptive information is also stored by the DBMS in the form of a database catalogue or dictionary; it is called meta-data

- An application program accesses the database by sending queries or requests for data to the DBMS. A query typically causes some data to be retrieved.

A transaction may cause some data to be read and some data to be written into the database.

Other important functions provided by the DBMS include protecting the database and maintaining it over a long period of time

- Protection includes system protection against hardware or software malfunction (or crashes) and security protection against unauthorized or malicious access
- A typical large database may have a life cycle of many years, so the DBMS must be able to maintain the database system by allowing the system to evolve as requirements change over time

CHARACTERISTICS OF THE DATABASE APPROACH

1. Self-describing nature of a database system: A fundamental characteristic of the database approach is that the database system contains not only the database itself but also a complete definition or description of the database structure and constraints.
   In other words, a self-describing database contains information about itself, allowing users and applications to understand the database's schema and content without external documentation.
2. Insulation between programs and data, and data abstraction

- The structure of data files is stored in the DBMS catalogue separately from the access programs. We call this property program data independence. Insulation is a clear separation between the programs (applications) that manipulate the data and the underlying data itself.
- Data abstraction refers to the process of hiding the complex internal details of data storage and organization from users and applications. The characteristic that allows program-data independence is called data abstraction.

3. Support of multiple views of the data: A multiuser DBMS whose users have a variety of distinct applications must provide facilities for defining multiple views.

- Support of multiple views of the data in a Database Management System (DBMS) refers to the ability of the system to present the same underlying data in different ways to different users or applications.

4. Sharing of data and multiuser transaction processing: A multiuser DBMS must allow multiple users to access the database at the same time. This is essential if data for multiple applications is to be integrated and maintained in a single database. A fundamental role of multiuser DBMS software is to ensure that concurrent transactions operate correctly and efficiently.

PURPOSE OF DATABASE SYSTEMS

1. Data Storage and Management: The primary purpose of a database system is to provide a structured and organized platform for storing vast amounts of data efficiently
2. Efficient Data Retrieval: Database systems enable quick and efficient data retrieval.
3. Data Sharing and Collaboration: Database systems facilitate data sharing and collaboration among multiple users and applications

In summary, database systems play a critical role in managing and organizing data, providing a foundation for data-driven applications, decision-making processes, and collaborative work environments. They ensure data efficiency, security, and integrity, making them indispensable tools in today's data-driven world.

Disadvantages of file-processing system

1. Data redundancy and inconsistency: same information may be duplicated in several places Since different programmers create the files and application programs over a long period.
2. Difficulty in accessing data
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
5. Atomicity problems
6. Security problems
7. Concurrent-access anomalies.

ADVANTAGES OF DBMS:

1. **Controlling of Redundancy**: Data redundancy refers to the duplication of data. In a database system, by having a centralized database and centralized control of data by the DBA the unnecessary duplication of data is avoided. It also eliminates the extra time for processing the large volume of data. It results in saving the storage space
2. **Improved Data Sharing**: DBMS allows a user to share the data in any number of application programs
3. **Data Integrity**: Integrity means that the data in the database is accurate. Centralized control of the data helps in permitting the administrator to define integrity constraints to the data in the database
4. **Security**: Having complete authority over the operational data, enables the DBA in ensuring that the only means of access to the database is through proper channels.
5. **Data Consistency**: By eliminating data redundancy, we greatly reduce the opportunities for inconsistency.
6. **Efficient Data Access**: In a database system, the data is managed by the DBMS and all access to the data is through the DBMS providing a key to effective data processing.
7. **Enforcements of Standards**: With the centralization of data, DBA can establish and enforce the data standards which may include the naming conventions, data quality standards etc
8. **Data Independence**: Ina database system, the database management system provides the interface between the application programs and the data.

DISADVANTAGES OF DBMS

1. It is bit complex**.** Since it supports multiple functionality to give the user the best, the underlying software has become complex.
2. it uses large amount of memory. It also needs large memory to run efficiently.
3. DBMS system works on the centralized system, i.e.; all the users from all over the world access this database. Hence any failure of the DBMS, will impact all the users.
4. DBMS is generalized software, i.e.; it is written work on the entire systems rather specific one. Hence some of the application will run slow.

FUNCTIONS OF DBMS

1. **Controlling Redundancy**: Controlling redundancy is a fundamental function of a Database Management System (DBMS) that aims to minimize data duplication and ensure data consistency within a database.

- **Data Normalization:** this involves organizing data into multiple related tables to eliminate data duplication and to maintain data integrity.
2. **Restricting Unauthorized Access**: A DBMS should provide a security and authorization subsystem, which the DBA uses to create accounts and to specify account restrictions.
- Restricting Unauthorized Access is a crucial function of a Database Management System (DBMS) to ensure that only authorized users can access and manipulate the data within a database. This function is essential for maintaining data security, privacy, and integrity
3. **Providing Backup and Recovery**: A DBMS must provide facilities for recovering from hardware or software failures. The backup and recovery subsystem of the DBMS is responsible for recovery.
4. **Providing Multiple User Interfaces**: a DBMS should provide a variety of user interfaces Because many types of users with varying levels of technical knowledge use a database. These include query languages for casual users, programming language interfaces for application programmers
5. **Representing Complex Relationships among Data:** A database may include numerous varieties of data that are interrelated in many ways. DBMS must have the capability to represent a variety of complex relationships among the data, to define new relationships as they arise, and to retrieve and update related data easily and efficiently
6. **Enforcing Integrity Constraints:** Enforcing Integrity Constraints is a critical function of a Database Management System (DBMS) that ensures the accuracy and consistency of data stored in a database. Integrity constraints are predefined rules and conditions that the data must adhere to, and the DBMS enforces these constraints to prevent invalid or inconsistent data from being entered into the database.

Database Users

1. **Database Administrators:** is A person who has central control over the system is called database administrator (DBA). The functions of DBA are:
- 1. Creation and modification of conceptual Schema definition
- 2. Implementation of storage structure and access method.
- 3. Schema and physical organization modifications
- 4. Granting of authorization for data access.
- 5. Integrity constraints specification.
- 6. Execute immediate recovery procedure in case of failures
- 7. Ensure physical security to database
2. **Database Designers:** Database designers are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data. It is the responsibility of database designers to communicate with all prospective database users in order to understand their requirements and to create a design that meets these requirements.
3. **End Users:** End users are the people whose jobs require access to the database for querying, updating, and generating reports; the database primarily exists for their use. There are several categories of end users:
- **Naive users** are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.

- **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 without writing a program.
- **Sophisticated users** interact with the system without writing programs. Instead, they form their requests in a database query language. They submit each such query to a query processor whose function is to break down DML statements into instructions that the storage manager understands. Analysts who submit queries to explore data in the database fall in this category

Data Models, Schemas and Instances

Data Model: A data model is a collection of concepts that can be used to describe the structure of a database that provides the necessary means to achieve data abstraction.

Categories of Data Models

1. High-level or conceptual data models provide concepts that are close to the way many users perceive data. it uses concepts such as entities, attributes, and relationships.
2. low-level or physical data models provide concepts that describe the details of how data is stored on the computer storage media, typically magnetic disks.
   Concepts provided by low-level data models are generally meant for computer specialists, not for end users

Database schema: a database schema defines how the data is organized and stored in the database. it is a logical representation or blueprint of the structure of a database.

A displayed schema is called a schema diagram. The following figure shows a schema diagram of a database. The diagram displays the structure of each record type but not the actual instances of records. We call each object in the schema—such as STUDENT or COURSE—a schema construct.

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

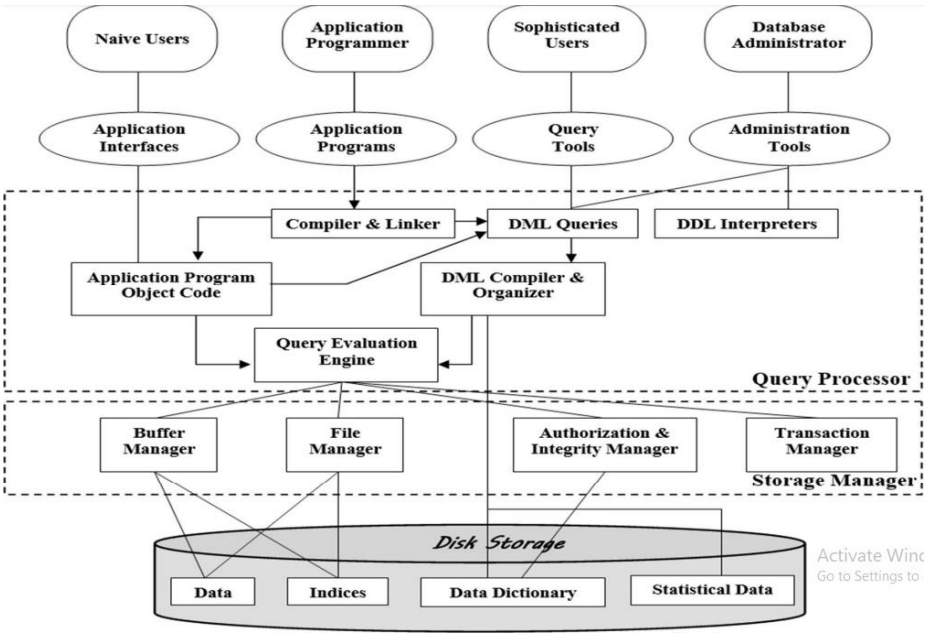
SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

DATABASE ARCHITECTURE

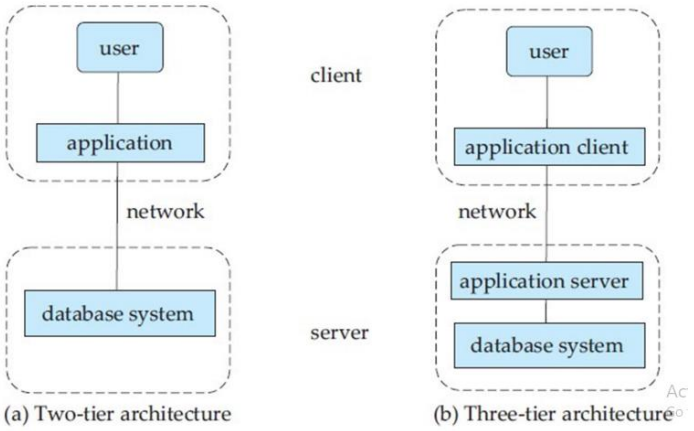


Typical Database Structure

1. **Query Processor**: The query processor components include:
   · **DDL interpreter**, which interprets DDL statements and records the definitions in the data dictionary.
   · **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.
   . **Query evaluation engine**, which executes low-level instructions generated by the DML compiler.
2. **Storage Manager**: A storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system. The storage manager is responsible for the interaction with the file manager. The storage manager components include:
   · Authorization and integrity manager, which tests for the satisfaction of integrity constraints and checks the authority of users to access data.
   · Transaction manager, which ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.
   · File manager, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.
   · Buffer manager, which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory.
3. **Transaction Manager:** A transaction is a collection of operations that performs a single logical function in a database application. Each transaction is a unit of both atomicity and consistency. Transaction - manager ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.

TWO TIER ARCHITECTURE: In a two-tier architecture, the application resides at the client machine, where it invokes database system functionality at the server machine through query language statements. Application program interface standards like ODBC and JDBC are used for interaction between the client and the server

THREE TIER ARCHITECTURE: in a three-tier architecture, the client machine acts as merely a front end and does not contain any direct database calls. Instead, the client end communicates with an application server, usually through a forms interface.



THREE-SCHEMA ARCHITECTURE AND DATA INDEPENDENCE

A database management system that provides three level of data is said to follow three-level architecture. The goal of the three-schema architecture, illustrated in Figure is to separate the user applications from the physical database. In this architecture, schemas can be defined at the following three levels:

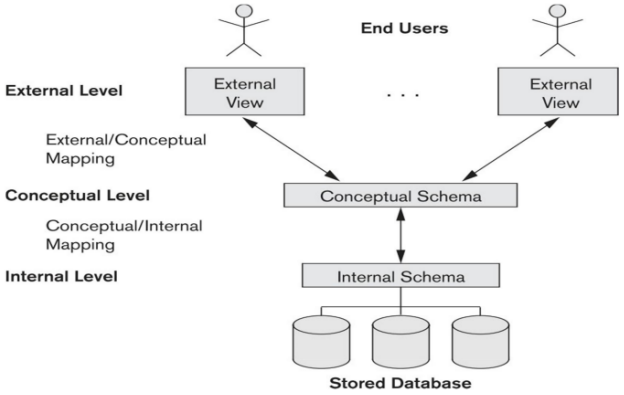
The goal of the three-schema architecture, illustrated in the above figure is to separate the user applications from the physical database. In this architecture, schemas can be defined at the following three levels:

- **External Level:** The external level is at the highest level of database abstraction. At this level, there will be many views defined for different user requirement. A view will describe only a subset of the database. Any number of user views may exist for a given global schema (conceptual schema). Thus, this level of abstraction is concerned with different categories of users. Each external view is described by means of a schema called sub schema.
- **Conceptual Level:** At this level of database abstraction all the database entities and the relationships among them are included. One conceptual view represents the entire database. This conceptual view is defined by the conceptual schema. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations and constraints.
- **Internal level:** It is the lowest level of abstraction closest to the physical storage method used. It indicates how the data will be stored and describes the data structures and access methods to be used by the database.

The internal view is expressed by internal schema. The following aspects are considered at this level:
1. Storage allocation example: B-tree, hashing

2. Access paths example specification of primary and secondary keys, indexes

3. Miscellaneous example Data compression and encryption techniques, optimization of the internal structures.



DATA INDEPENDENCE

data independence, which can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level. We can define two types of data independence:

1. **Logical data independence** is the capacity to change the conceptual schema without having to change external schemas or application programs. We may change the conceptual schema to expand the database (by adding a record type or data item), to change constraints, or to reduce the database
2. **Physical data independence** is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schemas need not be changed as well.

Generally, physical data independence exists in most databases and file environments where physical details such as the exact location of data on disk, and hardware details of storage encoding, placement, compression, splitting, merging of records, and so on are hidden from the user

On the other hand, logical data independence is harder to achieve because it allows structural and constraint changes without affecting application programs—a much stricter requirement.

OVERVIEW OF STORAGE AND INDEXING

1. Data on external storage:

Data in a DBMS is stored on storage devices such as disks and tapes

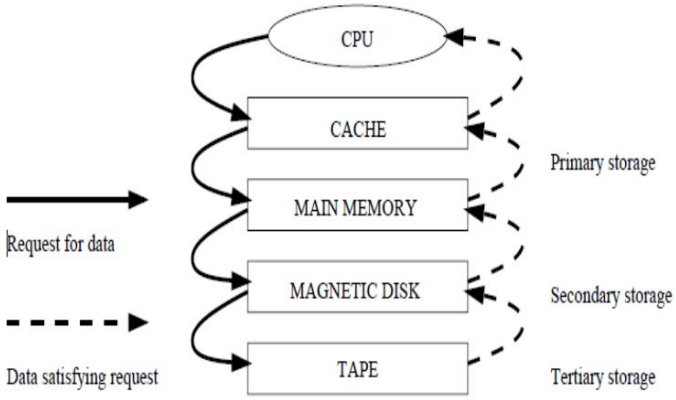
- The disk space manager is responsible for keeping track of available disk space.

• The file manager, which provides the abstraction of a file of records to higher levels of DBMS code, issues requests to the disk space manager to obtain and relinquish space on disk.

Storage Manager Component: is a program module that provides the interface between the low-level data stored in the database and the application programs/queries submitted to the system. The Storage Manager Components include –

- File Manager- File manager manages the file space and it takes care of the structure of the file
- Buffer Manager – It transfers blocks between disk and Main Memory
- Authorization and Integrity Manager – checks for the authority of the users to access and modify information, as well as integrity constraints
- Disk Manager- The block requested by the file manager is transferred by the Disk Manager

Memory Hierarchy:



CLASSIFICATION OF STORAGE DEVICES

- Primary Storage: is the main memory of a computer. At this level, the memory hierarchy include the cache memory which is a static RAM (cache memory is mainly used by the CPU to speedup execution programs.) and DRAM which provide the work area for the CPU for keeping programs and data
- the advantage of DRAM is it is low cost, which continuous to decrease;
- the drawback is its volatility and lower speed compared with static RAM.
- Secondary Storage: is external devices such as hard drives. At this level, the hierarchy includes magnetic disks, as well storage in the form of CD - ROM devices.
- Tertiary storage: at this level, the hierarchy includes optical disks and tapes as the least expensive end. The storage capacity anywhere in the hierarchy is measured in KB, MB, GB, TB

Access time: the time it takes from when a read or write request is issued to when data transfer begins.
Data-transfer rate– the rate at which data can be retrieved from or stored to the disk.
Mean time to failure (MTTF)– the average time the disk is expected to run continuously without any failure.

Explanation:

DRAM: programs reside execute in DRAM. Generally, large permanent database resides on secondary storage, and portions of the database are read into and written from buffers to main memory as needed.

Flash Memory: between DRAM and Magnetic disk, flash memory resides which is becoming common, particularly because it is non - volatile. It is high density, high - performance memory using EEPROM technology.

- the advantage of flash memory is the fast access speed
- the disadvantage is that an entire block must be erased and written over at a time.

Magnetic disk storage: is a primary medium for long-term storage. Data must be moved from disk to main memory in order for the data to be operated on.

• After operations are performed, data must be copied back to disk if any changes were made.

• Disk storage is called direct access storage as it is possible to read data on the disk in any order (unlike sequential access).

• Disk storage usually survives power failures and system crashes.

CD-ROM: store data optically and are read by a laser. CD - ROM contain pre - recorded data that cannot be overwritten

Tapes: are relatively not expensive and can store very large amount of data. when we maintain data for a long period but do expect to access it very often.it is Cheaper, but much slower access, since tape must be read sequentially from the beginning. And it is Used as protection from disk failures!

2. FILE ORGANIZATIONS: is the process of storing the files in certain order. It is the way in which data are organized within a file on a storage device.

The main objectives of file organization are:

- Optimal selection of records i.e.; records should be accessed as fast as possible.
- Any insert, update or delete transaction on records should be easy, quick and should not harm other records.
- No duplicate records should be induced as a result of insert, update or delete
- Records should be stored efficiently so that cost of storage is minimal.

Some of the file organizations are

1. Sequential File Organization
2. Heap File Organization
3. Hash/Direct File Organization
4. B+ Tree File Organization
5. Cluster File Organization

1. Sequential File Organization: is the easiest method for FO, in this, records are placed in the file in some sequential order based on the unique key field or search key. There are two ways to implement this method:

- Pile File Method – This method is simple, in which we store the records in a sequence that is one after other in the order in which they are inserted into the tables.
- Sorted File Method –In this method, whenever a new record has to be inserted, it is always inserted in a sorted (ascending or descending) manner.

Pros and Cons of Sequential File Organization –
Pros –

- Fast and efficient method for huge amount of data.
- Simple design.
- Files can be easily stored in magnetic tapes i.e cheaper storage mechanism.

Cons –

- Time wastage as we cannot jump on a particular record that is required, but we have to move in a sequential manner which takes our time.
- Sorted file method is inefficient as it takes time and space for sorting records.

2. Heap File Organization: When a file is created using Heap File Organization, the OS allocates memory area to that file without any further accounting details. File records can be placed anywhere in that memory area.

Heap File Organization works with data blocks. In this method records are inserted at the end of the file, into the data blocks. No Sorting or Ordering is required in this method.

Pros and Cons of Heap File Organization –
Pros –

- Fetching and retrieving records is faster than sequential record but only in case of small databases.
- When there is a huge number of data needs to be loaded into the database at a time, then this method of file Organization is best suited.

Cons –

- Problem of unused memory blocks.
- Inefficient for larger databases.

3. Hash File Organization: uses Hash function computation on some fields of the records. The output of the hash function determines the location of disk block where the records are to be placed.

In this method of file organization:

- hash function is used to calculate the address of the block to store the records.

- The hash function can be any simple or complex mathematical function
- The hash function is applied on some columns/attributes – either key or non-key columns to get the block address
- this method is also known as Direct or Random file organization.

Advantages of Hash File Organization

- Records need not be sorted after any of the transaction. Hence the effort of sorting is reduced in this method.
- Since block address is known by hash function, accessing any record is very faster. Similarly updating or deleting a record is also very quick.

- This method can handle multiple transactions as each record is independent of other. i.e.; since there is no dependency on storage location for each record, multiple records can be accessed at the same time.
- It is suitable for online transaction systems like online banking, ticket booking system

5. clustered file organization: is not considered good for large databases. In this mechanism, related records from one or more relations are kept in the same disk block, that is, the ordering of records is not based on primary key or search key.

REDUNDANT ARRAY OF INDEPENDENT DISKS(RAID)

is a technology to connect multiple secondary storage devices and use them as a single storage media. RAID levels define the use of disk arrays.

- RAID 0: In this level, a striped array of disks is implemented. The data is broken down into blocks and the blocks are distributed among disks.
- RAID 1: uses mirroring techniques. When data is sent to a RAID controller, it sends a copy of data to all the disks in the array
- RAID 2: records Error Correction Code using Hamming distance for its data, striped on different disks
- RAID 3: stripes the data onto multiple disks. The parity bit generated for data word is stored on a different disk. This technique makes it to overcome single disk failures
- RAID 4: In this level, an entire block of data is written onto data disks and then the parity is generated and stored on a different disk.
- RAID 5 writes whole data blocks onto different disks, but the parity bits generated for data block stripe are distributed among all the data disks rather than storing them on a different dedicated disk.
- RAID 6 is an extension of level 5. In this level, two independent parities are generated and stored in distributed fashion among multiple disks

The operations to be considered for comparisons of file organizations are below:

- **Scan:** Fetch all records in the file. The pages in the file must be fetched from disk into the buffer pool. There is also a CPU overhead per record for locating the record on the page (in the pool).
- **Search with equality selection:** Fetch all records that satisfy an equality selection, for example, "Find the Students record for the student with *sid* 23." Pages that contain qualifying records must be fetched from disk, and qualifying records must be located within retrieved pages.
- **Search with range selection:** Fetch all records that satisfy a range selection, for example, "Find all Students records with *name* alphabetically after 'Smith.' "
- **Insert:** Insert a given record into the file. We must identify the page in the file into which the new record must be inserted, fetch that page from disk, modify it to include the new record, and then write back the modified page. Depending on the file organization, we may have to fetch, modify, and write back other pages as well.
- **Delete:** Delete a record that is specified using its rid. We must identify the page that contains the record, fetch it from disk, modify it, and write it back. Depending on the file organization, we may have to fetch, modify, and write back other pages as well.

File Type	*Scan*	*Equality Search*	*Range Search*	*Insert*	*Delete*
Heap	BD	$0.5BD$	BD	$2D$	$Search + D$
Sorted	BD	$Dlog_2B$	$Dlog_2B$+# *matches*	$Search + BD$	$Search + BD$
Hashed	$1.25BD$	D	$1.25BD$	$2D$	$Search + D$

INDEXING: is a data structure technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done. Indexing can be of the following types:

• Primary Index: is defined on an ordered data file. The data file is ordered on a key field. The key field is generally the primary key of the relation.

• Secondary Index: may be generated from a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.

• Clustering Index: is defined on an ordered data file. The data file is ordered on a non-key field.

Ordered Indexing is of two types

- **Dense:** In dense index, there is an index record for every search key value in the database. This makes searching faster but requires more space to store index records itself.
- **Sparse:** In sparse index, index records are not created for every search key. An index record here contains a search key and an actual pointer to the data on the disk.

Multilevel Index: is stored on the disk along with the actual database files. As the size of the database grows, so does the size of the indices

B^+ tree is a balanced binary search tree that follows a multi-level index format. The leaf nodes of a B+ tree denote actual data pointers

RELATIONAL MODEL

is simple model in which database is represented as a collection of “relations” where each relation is represented by two-dimensional table

Properties:

- It is column homogeneous.
- All rows of a table are distinct.
- The ordering of rows with in a table is immaterial.
- Each item is a simple number or a character string
- The column of a table is assigned distinct names and the ordering of these columns is immaterial.

Tuple: is a single row of data in a table, representing a complete record with ordered attributes.

Attributes: is a named column within a table that holds specific data about each record or tuple in the table.

Domain: is a set of values that can be given to an attribute. every attribute in a table has a specific domain. Values to these attributes could not be assigned outside their domains.

Relation: A relation consist of **Relational schema** and **Relation instance**

- **Relational Schema:** A relational schema specifies the relation’s name, its attributes and the domain of each attribute.
- **Relation Instance:** is a collection of tuples for a given relational schema at a specific point of time.

Keys: refers to one or more columns within a table that uniquely identify each row in that table.

Super key: is a set of attributes used to identify the records uniquely in a relation.

Candidate keys: are minimal super keys. candidate key is a combination of attributes that could potentially serve as a primary key because it uniquely identifies rows.

Primary key: is the candidate key that is chosen by the database designer as the principal means of identifying entities within an entity set.

CONSTRAINTS

There are three types of constraints on relational database that include

- domain constraints
- key constraints
- integrity constraints

1. domain constraints specify the condition that we to put on each instance of the relation. So, the values that appear in each column must be drawn from the domain associated with that column.
2. Key Constraints: This constraints states that the key attribute value in each tuple must be unique
3. Integrity CONSTRAINTS: There are two types of integrity constraints:
    - Entity Integrity Constraints
    - Referential Integrity constraints

Entity Integrity Constraints: It states that no primary key value can be null and unique. This is because the primary key is used to identify individual tuple in the relation.

Referential Integrity Constraints: It states that the tuple in one relation that refers to another relation must refer to an existing tuple in that relation.