Unit 6: Trends in Database Technology

Storage System in DBMS

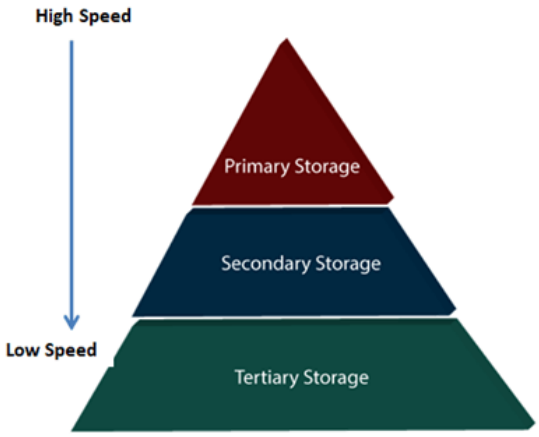
A database system provides an ultimate view of the stored data. However, data in the form of bits, bytes get stored in different storage devices.

In this section, we will take an overview of various types of storage devices that are used for accessing and storing data.

Types of Data Storage

For storing the data, there are different types of storage options available. These storage types differ from one another as per the speed and accessibility. There are the following types of storage devices used for storing the data:

- Primary Storage
- Secondary Storage
- Tertiary Storage



Primary Storage

It is the primary area that offers quick access to the stored data. We also know the primary storage as volatile storage. It is because this type of memory does not permanently store the data. As soon as the system leads to a power cut or a crash, the data also gets lost. Main memory and cache are the types of primary storage.

Main Memory: It is the one that is responsible for operating the data that is available by the storage medium. The main memory handles each instruction of a computer machine. This type of memory can store gigabytes of data on a system but is small enough to carry the entire database. At last, the main memory loses the whole content if the system shuts down because of power failure or other reasons.

Cache: It is one of the costly storage media. On the other hand, it is the fastest one. A cache is a tiny storage media which is maintained by the computer hardware usually. While designing the algorithms and query processors for the data structures, the designers keep concern on the cache effects.

Secondary Storage

Secondary storage is also called Online storage. It is the storage area that allows the user to save and store data permanently. This type of memory does not lose the data due to any power failure or system crash. That's why we also call it non-volatile storage.

There are some commonly described secondary storage media which are available in almost every type of computer system:

- **Flash Memory:** A flash memory stores data in USB (Universal Serial Bus) keys which are further plugged into the USB slots of a computer system. These USB keys help transfer data to a computer system, but it varies in size limits. Unlike the main memory, it is possible to get back the stored data which may be lost due to a power cut or other reasons. This type of memory storage is most commonly used in the server systems for caching the frequently used data. This leads the systems towards high performance and is capable of storing larger amounts of databases than the main memory.
- **Magnetic Disk Storage:** This type of storage media is also known as online storage media. A magnetic disk is used for storing the data for a long time. It is capable of storing an entire database. It is the responsibility of the computer system to make availability of the data from a disk to the main memory for further accessing. Also, if the system performs any operation over the data, the modified data should be written back to the disk. The tremendous capability of a magnetic disk is that it does not affect the data due to a system crash or failure, but a disk failure can easily ruin as well as destroy the stored data.

Tertiary Storage

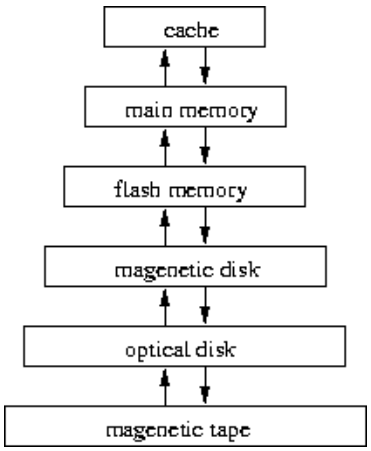
It is the storage type that is external from the computer system. It has the slowest speed. But it is capable of storing a large amount of data. It is also known as Offline storage. Tertiary storage is generally used for data backup. There are following tertiary storage devices available:

- **Optical Storage:** An optical storage can store megabytes or gigabytes of data. A Compact Disk (CD) can store 700 megabytes of data with a playtime of around 80 minutes. On the other hand, a Digital Video Disk or a DVD can store 4.7 or 8.5 gigabytes of data on each side of the disk.
- **Tape Storage:** It is the cheapest storage medium than disks. Generally, tapes are used for archiving or backing up the data. It provides slow access to data as it accesses data sequentially from the start. Thus, tape storage is also known as sequential-access storage. Disk storage is known as direct-access storage as we can directly access the data from any location on disk.

Storage Hierarchy

Besides the above, various other storage devices reside in the computer system. These storage media are organized on the basis of data accessing speed, cost per unit of data to buy the medium, and by medium's reliability. Thus, we can create a hierarchy of storage media on the basis of its cost and speed.

Thus, on arranging the above-described storage media in a hierarchy according to its speed and cost, we conclude the below-described image:



In the image, the higher levels are expensive but fast. On moving down, the cost per bit is decreasing, and the access time is increasing. Also, the storage media from the main memory to up represents the volatile nature, and below the main memory, all are non-volatile devices.

RAID

RAID refers to the redundancy array of the independent disk. It is a technology which is used to connect multiple secondary storage devices for increased performance, data redundancy or both. It gives you the ability to survive one or more drive failures depending upon the RAID level used.

It consists of an array of disks in which multiple disks are connected to achieve different goals.

RAID technology

There are 7 levels of RAID schemes. These schemas are as RAID 0, RAID 1, ..., RAID 6.

These levels contain the following characteristics:

- It contains a set of physical disk drives.
- In this technology, the operating system views these separate disks as a single logical disk.
- In this technology, data is distributed across the physical drives of the array.
- Redundancy disk capacity is used to store parity information.
- In case of disk failure, the parity information can be helped to recover the data.

Standard RAID levels

RAID 0

- RAID level 0 provides data stripping, i.e., data can be placed across multiple disks. It is based on stripping that means if one disk fails then all data in the array is lost.
- This level doesn't provide fault tolerance but increases the system performance.

Disk 0	Disk 1	Disk 2	Disk 3
20	21	22	23
24	25	26	27
28	29	30	31
32	33	34	35

In this figure, block 0, 1, 2, 3 form a stripe.

In this level, instead of placing just one block into a disk at a time, we can work with two or more blocks placed into a disk before moving on to the next one.

Disk 0	Disk 1	Disk 2	Disk 3
20	22	24	26
21	23	25	27
28	30	32	34
29	31	33	35

In this above figure, there is no duplication of data. Hence, a block once lost cannot be recovered.

RAID 1

This level is called mirroring of data as it copies the data from drive 1 to drive 2. It provides 100% redundancy in case of a failure.

Example:

Disk 0	Disk 1	Disk 2	Disk 3
A	A	B	B
C	C	D	D
E	E	F	F
G	G	H	H

Only half space of the drive is used to store the data. The other half of the drive is just a mirror to the already stored data.

RAID 3

RAID 3 implements byte-level striping of Data. Data is stored across disks with their parity bits in a separate disk. The parity bits help to reconstruct the data when there is a data loss.

Let's see RAID 3 High-level implementation with an example:

DISK 0	DISK 1	DISK 2	DISK 3
10	11	12	P(10,11,12)
14	15	16	P(14,15,16)
18	19	20	P(18,19,20)
22	23	24	P(22,23,24)

Here Disk 3 contains the Parity bits for Disk 0 Disk 1 and Disk 2. If any one of the Disk's data is lost the data can be reconstructed using parity bits in Disk 3.

RAID 4

RAID 4 implements block-level striping of data with dedicated parity drive. If only one of the data is lost in any disk then it can be reconstructed with the help of parity drive. Parity is calculated with the help of XOR operation over each data disk block.

Let's see RAID 4 with an example:

DISK 0	DISK 1	DISK 2	DISK 3
0	1	0	P0
1	1	0	P1

Here P0 is calculated using XOR(0,1,0) = 1 and P1 is calculated using XOR(1,1,0) = 0 If there is an even number of 1 then XOR is 0 and for an odd number of 1 XOR is 1. If suppose Disk 0 data is lost, by checking parity P0=1 we will know that Disk 0 should have 0 to make the Parity P0 as 1 whereas if there was 1 in Disk 0 it would have made the parity P0=0 which contradicts with the current parity value.

RAID 5

RAID 5 is similar to RAID 4 with only one difference. The parity Rotates among the Disks.

Let's see an example of RAID 5 implementation:

DISK 0	DISK 1	DISK 2	DISK 3
0	1	0	P0
1	1	P1	0
1	P2	0	1
P3	1	0	0

Here We can see the rotation of Parity bits from Disk 3 to Disk 1.

RAID 6

If there is more than one Disk failure, then RAID 6 implementation helps in that case. In RAID 6 there are two parity in each array/row. It is similar to RAID 5 with extra parity.

Let's See RAID 6 with the help of an example :

DISK 0	DISK 1	DISK 2	DISK 3
0	1	Q0	P0
1	Q1	P1	0
Q2	P2	0	1
P3	1	0	Q3

Here P0,P1,P2,P3 and Q0,Q1,Q2,Q3 are two parity to reconstruct the data if at most two disks fail

.

File Organization

- The **File** is a collection of records. Using the primary key, we can access the records. The type and frequency of access can be determined by the type of file organization which was used for a given set of records.
- File organization is a logical relationship among various records. This method defines how file records are mapped onto disk blocks.
- File organization is used to describe the way in which the records are stored in terms of blocks, and the blocks are placed on the storage medium.
- The first approach to map the database to the file is to use several files and store only one fixed length record in any given file. An alternative approach is to structure our files so that we can contain multiple lengths for records.
- Files of fixed length records are easier to implement than the files of variable length records.

Objective of file organization

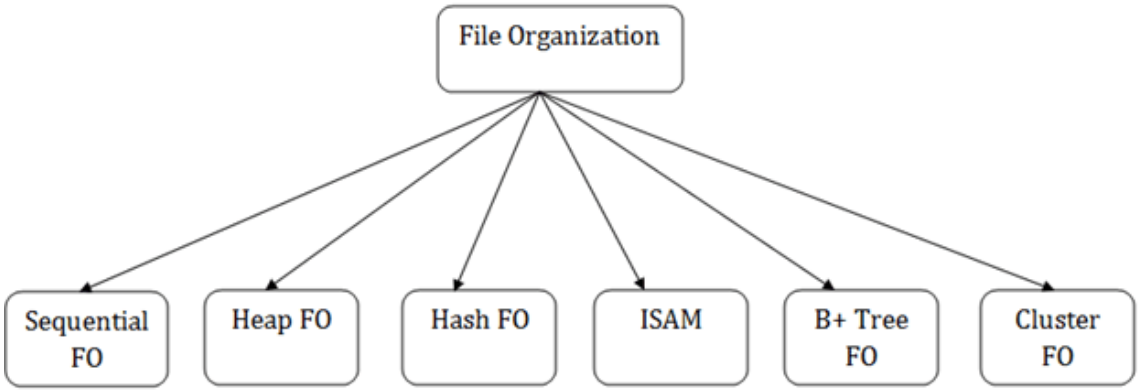
- It contains an optimal selection of records, i.e., records can be selected as fast as possible.
- To perform insert, delete or update transactions on the records should be quick and easy.

- The duplicate records cannot be induced as a result of insert, update or delete.
- For the minimal cost of storage, records should be stored efficiently.

Types of file organization:

File organization contains various methods. These particular methods have pros and cons on the basis of access or selection. In the file organization, the programmer decides the best-suited file organization method according to his requirement.

Types of file organization are as follows:



Indexing in DBMS

- Indexing is used to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed.
- The index is a type of data structure. It is used to locate and access the data in a database table quickly.

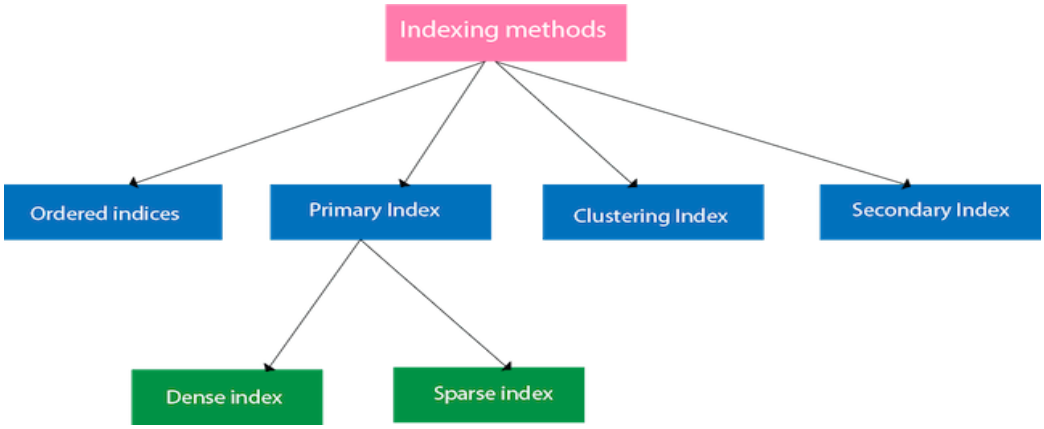
Index structure:

Indexes can be created using some database columns.

Search key	Data Reference

Fig: Structure of Index

- The first column of the database is the search key that contains a copy of the primary key or candidate key of the table. The values of the primary key are stored in sorted order so that the corresponding data can be accessed easily.
- The second column of the database is the data reference. It contains a set of pointers holding the address of the disk block where the value of the particular key can be found.



Ordered indices

The indices are usually sorted to make searching faster. The indices which are sorted are known as ordered indices.

Example: Suppose we have an employee table with thousands of records and each of which is 10 bytes long. If their IDs start with 1, 2, 3....and so on, we have to search students with ID-543.

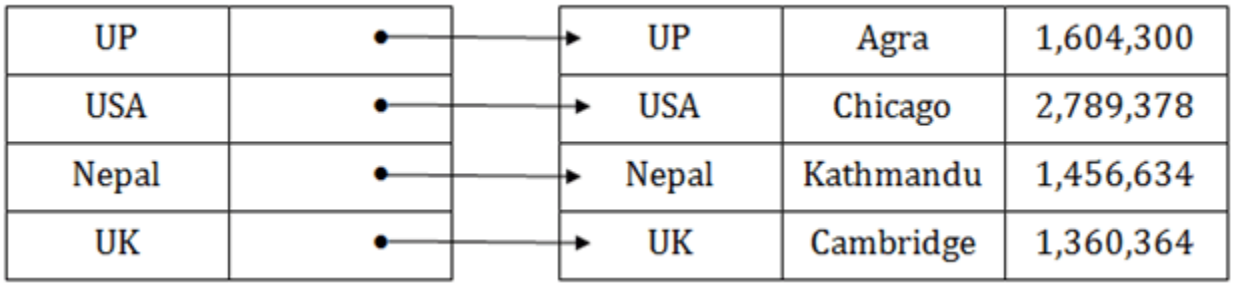
- In the case of a database with no index, we have to search the disk block from starting till it reaches 543. The DBMS will read the record after reading 543*10=5430 bytes.
- In the case of an index, we will search using indexes and the DBMS will read the record after reading 542*2= 1084 bytes which are very less compared to the previous case.

Primary Index

- If the index is created on the basis of the primary key of the table, then it is known as primary indexing. These primary keys are unique to each record and contain 1:1 relation between the records.
- As primary keys are stored in sorted order, the performance of the searching operation is quite efficient.
- The primary index can be classified into two types: Dense index and Sparse index.

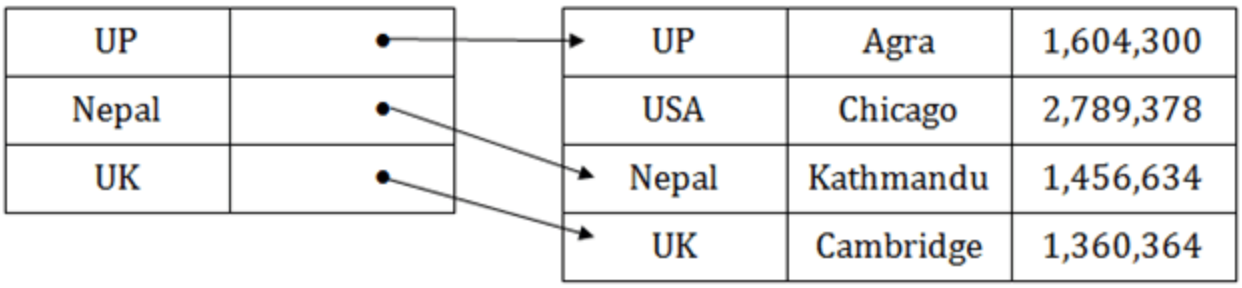
Dense index

- The dense index contains an index record for every search key value in the data file. It makes searching faster.
- In this, the number of records in the index table is the same as the number of records in the main table.
- It needs more space to store the index record itself. The index records have the search key and a pointer to the actual record on the disk.



Sparse index

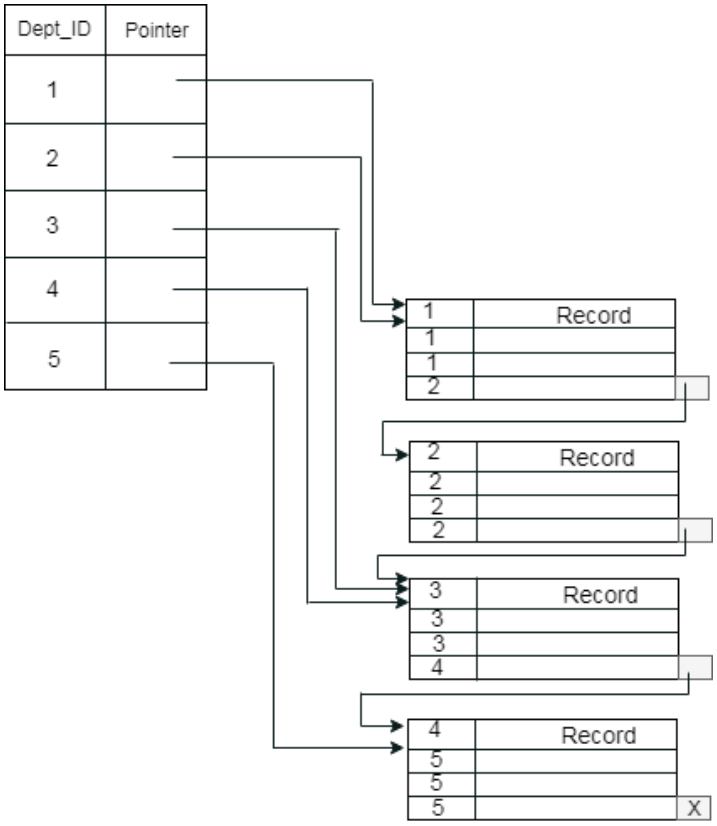
- In the data file, the index record appears only for a few items. Each item points to a block.
- In this, instead of pointing to each record in the main table, the index points to the records in the main table in a gap.



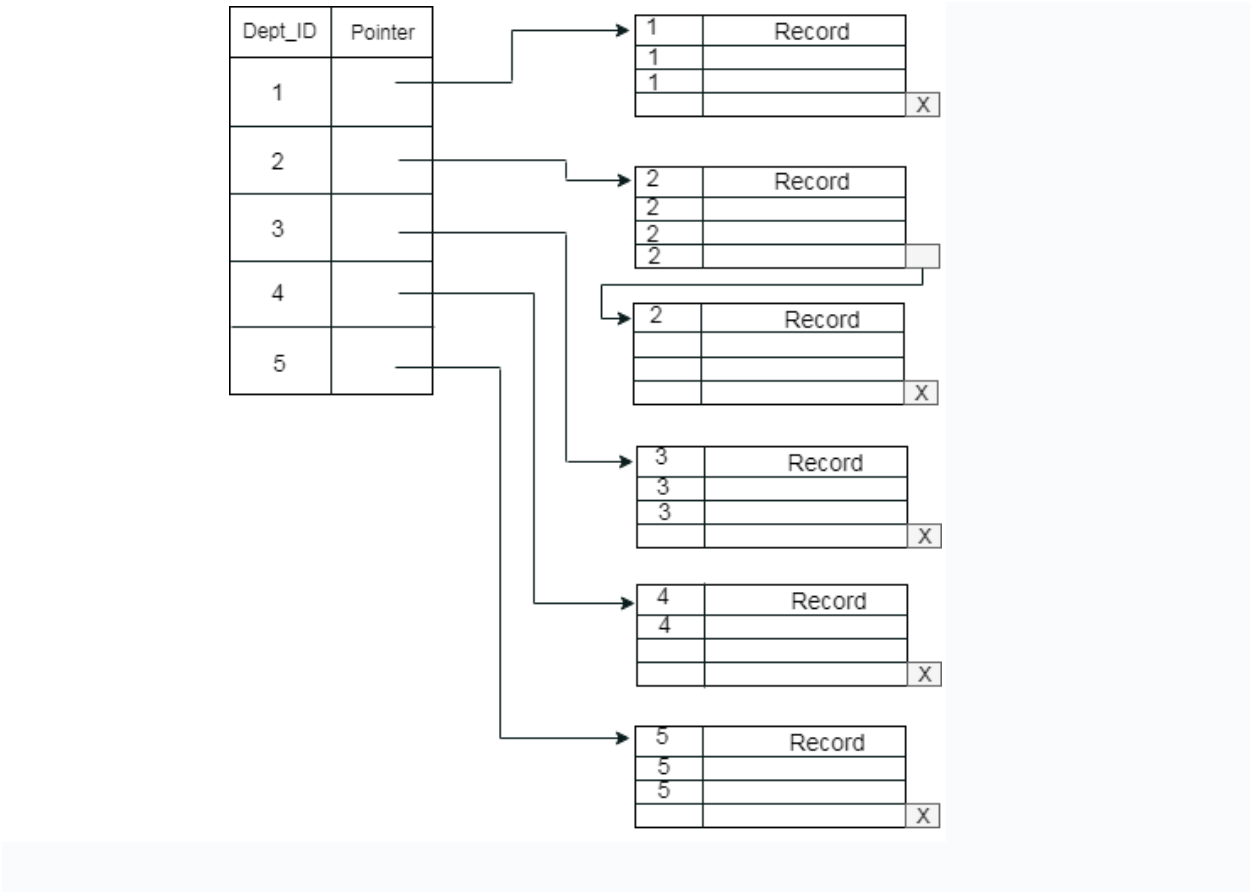
Clustering Index

- A clustered index can be defined as an ordered data file. Sometimes the index is created on non-primary key columns which may not be unique for each record.
- In this case, to identify the record faster, we will group two or more columns to get the unique value and create an index out of them. This method is called a clustering index.
- The records which have similar characteristics are grouped, and indexes are created for these groups.

Example: suppose a company contains several employees in each department. Suppose we use a clustering index, where all employees which belong to the same Dept_ID are considered within a single cluster, and index pointers point to the cluster as a whole. Here Dept_Id is a non-unique key.



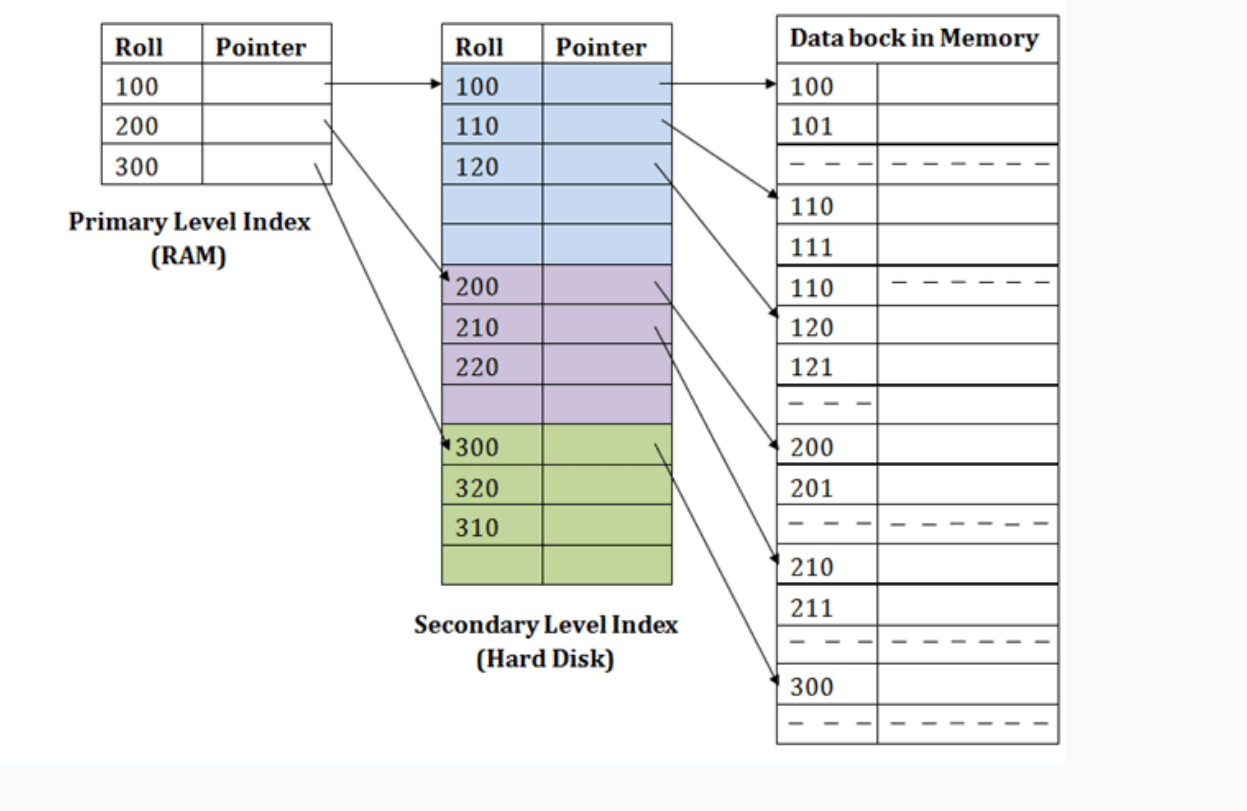
The previous schema is a little confusing because one disk block is shared by records which belong to the different cluster. If we use separate disk block for separate clusters, then it is called a better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address obtained from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk).



For example:

- If you want to find the record of roll 111 in the diagram, then it will search the highest entry which is smaller than or equal to 111 in the first level index. It will get 100 at this level.
- Then in the second index level, again it does max (111) <= 111 and gets 110. Now using the address 110, it goes to the data block and starts searching each record till it gets 111.
- This is how a search is performed in this method. Inserting, updating or deleting is also done in the same manner.

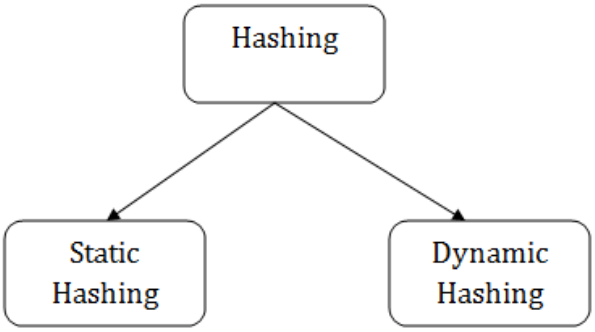
Hashing in DBMS

In a huge database structure, it is very inefficient to search all the index values and reach the desired data. Hashing technique is used to calculate the direct location of a data record on the disk without using index structure.

In this technique, data is stored at the data blocks whose address is generated by using the hashing function. The memory location where these records are stored is known as data buckets or data blocks.

In this, a hash function can choose any of the column values to generate the address. Most of the time, the hash function uses the primary key to generate the address of the data block. A hash function is a simple mathematical function to any complex mathematical function. We can even consider the primary key itself as the address of the data block. That means each row whose address will be the same as a primary key stored in the data block.

Types of Hashing:



Static Hashing

It is a hashing technique that enables users to lookup a definite data set. Meaning, the data in the directory is not changing, it is "Static" or fixed. In this hashing technique, the resulting number of data buckets in memory remains constant.

Operations Provided by Static Hashing

Static hashing provides the following operations –

Delete – Search a record address and delete a record at the same address or delete a chunk of records from records for that address in memory.

insertion – While entering a new record using static hashing, the hash function (h) calculates bucket address "h(K)" for the search key (k), where the record is going to be stored.

Search − A record can be obtained using a hash function by locating the address of the bucket where the data is stored.

Update − It supports updating a record once it is traced in the data bucket.

Static hashing is further divided into

1. Open hashing
2. Close hashing.

Advantages of Static Hashing

Static hashing is advantageous in the following ways −

Offers unparalleled performance for small-size databases.
Allows Primary Key value to be used as a Hash Key.

Disadvantages of Static Hashing

Static hashing comes with the following disadvantages −

It cannot work efficiently with the databases that can be scaled.
It is not a good option for large-size databases.
Bucket overflow issue occurs if there is more data and less memory.

Dynamic Hashing

It is a hashing technique that enables users to lookup a dynamic data set. Means, the data set is modified by adding data to or removing the data from, on demand hence the name 'Dynamic' hashing. Thus, the resulting data bucket keeps increasing or decreasing depending on the number of records.

In this hashing technique, the resulting number of data buckets in memory is ever-changing.

Operations Provided by Dynamic Hashing

Dynamic hashing provides the following operations −

Delete − Locate the desired location and support deleting data (or a chunk of data) at that location.
Insertion − Support inserting new data into the data bucket if there is a space available in the data bucket.
Query − Perform querying to compute the bucket address.
Update − Perform a query to update the data.

Advantages of Dynamic Hashing

Dynamic hashing is advantageous in the following ways –

- ➢ It works well with scalable data.
- ➢ It can handle addressing large amounts of memory in which data size is always changing.
- ➢ Bucket overflow issues come rarely or very late.

Disadvantages of Dynamic Hashing

Dynamic hashing comes with the following disadvantage –

The location of the data in memory keeps changing according to the bucket size. Hence if there is a phenomenal increase in data, then maintaining the bucket address table becomes a challenge.

Differences between Static and Dynamic Hashing

Key Factor	Static Hashing	Dynamic Hashing
Form of Data	Fixed-size, non-changing data.	Variable-size, changing data.
Result	The resulting Data Bucket is of fixed-length.	The resulting Data Bucket is of variable-length.
Bucket Overflow	Challenge of Bucket overflow can arise often depending upon memory size.	Bucket overflow can occur very late or doesn't occur at all.
Complexity	Simple	Complex

Introduction to Distributed Databases

Distributed database system is a type of database management system that stores data across multiple computers or sites that are connected by a network.

A distributed database(DDB) is the collection of the multiple ,logically interrelated database distributed over a computer network.

A distributed database management system(D-DBMS) is the software that manages the DDB and provides the access mechanism that makes this distribution transparent to the users.

Types:

1. **Homogeneous Database:** A homogeneous database stores data uniformly across all locations. All sites utilize the same operating system, database management system, and data structures. They are therefore simple to handle.

2. **Heterogeneous Database:** With a heterogeneous distributed database, many locations may employ various software and schema, which may cause issues with queries and transactions. Moreover, one site could not be even aware of the existence of the other sites. Various operating systems and database applications may be used by various machines. They could even employ separate database data models. Translations are therefore necessary for communication across various sites.

Data may be stored on several places in two ways using distributed data storage:

1. **Replication -** With this strategy, every aspect of the connection is redundantly kept at two or more locations. It is a completely redundant database if the entire database is accessible from every location. Systems preserve copies of the data as a result of replication. This has advantages since it makes more data accessible at many locations. Moreover, query requests can now be handled in parallel. But, there are some drawbacks as well. Data must be updated often. All changes performed at one site must be documented at every site where that relation is stored in order to avoid inconsistent results. There is a tone of overhead here. Moreover, since concurrent access must now be monitored across several sites, concurrency management becomes far more complicated.
2. **Fragmentation -** In this method, the relationships are broken up into smaller pieces and each fragment is kept in the many locations where it is needed. To ensure there is no data loss, the pieces must be created in a way that allows for the reconstruction of the original relation. As fragmentation doesn't result in duplicate data, consistency is not a concern.

Relationships can be fragmented in one of two ways:

- Separating the relation into groups of tuples using rows results in horizontal fragmentation, where each tuple is allocated to at least one fragment.
- Vertical fragmentation, also known as splitting by columns, occurs when a relation's schema is split up into smaller schemas. A common candidate key must be present in each fragment in order to guarantee a lossless join

Sometimes a strategy that combines fragmentation and replication is employed.

Multidimensional Databases

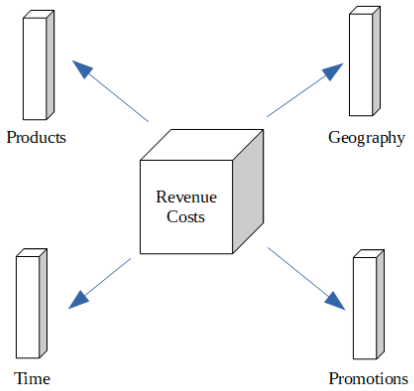
Multidimensional databases are used mostly for OLAP (online analytical processing) and data warehousing. They can be used to show multiple dimensions of data to users .

A multidimensional database is created from multiple relational databases. While relational databases allow users to access data in the form of queries, the multidimensional databases allow users to ask analytical questions related to business or market trends.

The multidimensional database uses MOLAP (multidimensional online analytical processing) to access its data. They allow the users to quickly get answers to their requests by generating and analyzing the data rather quickly.

The data in multidimensional databases is stored in a data cube format. This means that data can be seen and understood from many dimensions and perspectives.

Example



The revenue costs for a company can be understood and analyzed on the basis of various factors like the company products, the geographical locations of the company offices, time to develop a product, promotions done etc.

Parallel Databases

Parallel DBMS is a Database Management System that runs through multiple processors and disks. They combine two or more processors and disk storage that helps make operations and executions easier and faster. They are designed to execute concurrent operations. They exist, happen, or done at the same time even if the data processed are not from one source or one processing unit.
The main architecture for parallel DBMS is:

1. Shared Memory System

A Shared Memory System is an architecture of Database Management System, where every computer processor is able to access and process data from multiple memory modules or units through an intercommunication channel. This architecture is also commonly known as SMP or Symmetric Multi-processing. A Shared Memory System contains large amounts of cache memories at each processor, so referencing of the shared memory is avoided.

Advantages of Shared Memory:

Data is easily accessed from various processors.
A single processor can send messages to other processors efficiently.

Disadvantage of Shared Memory:

Waiting time for every single processor increases when all of them are used.
Bandwidth is also a problem.

2. Shared Disk System

A Shared Disk System is an architecture of Database Management System where every computer processor can access multiple disks through an intercommunication network. It can also access and utilize every local memory. Each of the processors have their own memory system, so the shared data is more efficient.

Advantages of Shared Disk System:

The fault tolerance can be achieved using this system.

Disadvantages of Shared Disk System:

The addition of processors can slow down existing processors.

Shared Disk Systems have a limited scalability meaning it is sometimes fixed.

3. Shared Nothing System

A Shared Nothing System is an architecture of Database Management System where every processor has its own disk and memory for the objective of efficient workflows. The processors can communicate with other processors using an intercommunication network. Each of the processors act like servers to store data on the disk. So there can be efficient and effective workflows.

Advantage of Shared Nothing System

This system has more scalability.Number of processors and disk can be connected as per the requirement in the share nothing disk system.

Disadvantage of Shared Nothing system

Must require the partitioning of data.
Cost that is needed for this system is higher.

a. Advantages of parallel DBMS :

Speed
The first advantage of parallel DBMS is speed. The servers from parallel DBMS are able to break up user database requests into parts and it dispatches each of the parts to separate computers. They work on these “work parts” simultaneously and they merge the results, passing them back to the user. This speeds up most of the data requests, allowing faster access to very large databases.
Reliability
The second advantage is reliability. A parallel database, when properly configured, can continue to work despite the failure of any computer in the cluster. The database server can sense that a specific computer is not responding, and it can reroute its work to the remaining computers.

Capacity
The third advantage is capacity. As more and more users request access to the database, the computer administrators add more computers to the parallel

server, boosting its overall capacity to the max, databases are more likely to slow down and have slower systems. A parallel database, for example, allows a large online retailer to have thousands of requests of users accessing information at the same time. This level of processing performance is not possible with single server systems.

b. Disadvantage of Parallel DBMS :

Cost

The first disadvantage of parallel DBMS is cost. As the need for quicker processing and efficient searches increase, there need to be more and more disks and processors that simultaneously work together to achieve the best and quickest results. To do that, Parallel DBMS needs lots of processors and disks in the first place. In the end, it is never cheap to implement parallel DBMSs.

Resources

The second disadvantage of parallel DBMS is the huge amount of resources. It is not and will never be easy to keep up with modern, cheap, and efficient resources. Implementing DBMS requires the users or the company to have renewal of resources, changing or maintaining resources, or even replacement of resources.

Difficulty of Managing Systems

The third disadvantage of parallel DBMS is the difficulty in managing systems. When having lots of systems, lots of resources, and lots of systems running it is never easy to manage them. When there needs to be a software update, a replacement, or maintenance that all of the system needs to do, it will be time consuming and resource consuming.

Spatial and multimedia databases

Spatial Database

Spatial data is associated with geographic locations such as cities,towns etc. A spatial database is optimized to store and query data representing objects. These are the objects which are defined in a geometric space.

Characteristics of Spatial Database

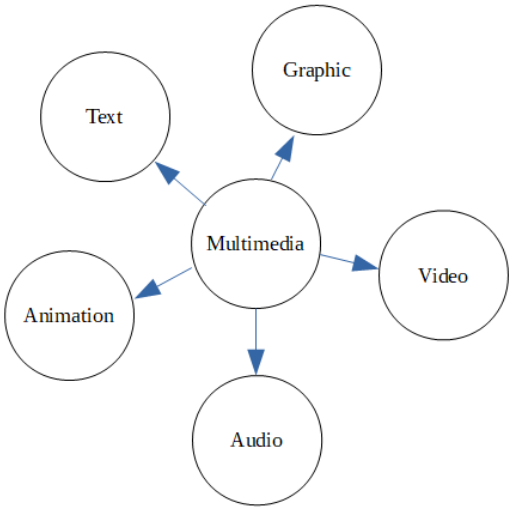
A spatial database system has the following characteristics

It is a database system

It offers spatial data types (SDTs) in its data model and query language.
It supports spatial data types in its implementation, providing at least spatial indexing and efficient algorithms for spatial join.

Multimedia Databases

The multimedia databases are used to store multimedia data such as images, animation, audio, video along with text. This data is stored in the form of multiple file types like .txt(text), .jpg(images), .swf(videos), .mp3(audio) etc.



Contents of the Multimedia Database:

The multimedia database stored the multimedia data and information related to it. This is given in detail as follows –

Media data

This is the multimedia data that is stored in the database such as images, videos, audios, animation etc.

Media format data

The Media format data contains the formatting information related to the media data such as sampling rate, frame rate, encoding scheme etc.

Media keyword data

This contains the keyword data related to the media in the database. For an image the keyword data can be date and time of the image, description of the image etc.

Media feature data

The Media feature data describes the features of the media data. For an image, feature data can be colors of the image, textures in the image etc.

Unit 7: Advanced Topic

Object-Oriented Database

An object-oriented database is a type of database that is based on the principles of object-oriented programming (OOP). In an object-oriented database, data is organized and stored as objects, which are self-contained units that contain both data and the operations or methods that can be performed on that data. This allows for the efficient representation and management of complex data structures and relationships.

Object-oriented databases are often used in applications that require the efficient management of complex data structures and relationships, such as CAD/CAM systems, geographic information systems, and document management systems. They are also well suited for applications that require the integration of different data types and sources, such as multimedia data or data from multiple sources. However, object-oriented databases can be more difficult to learn and use compared to other database models, and may require specialized expertise to set up and manage.

components of Object-Oriented Database Model

The main components of an object-oriented database model are:

Objects

In an object-oriented database model, data is organized and stored as objects, which are self-contained units that contain both data and the operations or methods that can be performed on that data.

Classes

Objects in an object-oriented database model are organized into classes, which define the properties and behavior of the objects. Classes can inherit properties and behavior from other classes, which allows for the efficient reuse of code and data structures.

Inheritance

Inheritance is a key concept in object-oriented database models, which allows classes to inherit properties and behavior from other classes. This allows for the efficient reuse of code and data structures, and simplifies the development and management of complex data structures.

Polymorphism

Polymorphism is another key concept in object-oriented database models, which allows objects to take on different forms or behaviors depending on the context in which they are used. This allows for the efficient representation and manipulation of complex data structures and relationships.

Persistence

In an object-oriented database model, objects are persistent, which means that they are stored in the database and can be accessed and manipulated by applications and users. This allows for the efficient management and manipulation of complex data structures and relationships.

What is the Difference Between a Relational Database and Object-Oriented Database

There are several key differences between a relational database and an object-oriented database, including the following:

1. **Data organization:** In a relational database, data is organized and stored in tables, with each table containing rows and columns of data. In an object-oriented database, data is organized and stored as objects, which are self-contained units that contain both data and the operations or methods that can be performed on that data.

2. **Data relationships:** In a relational database, data relationships are defined and managed using keys and foreign keys, which link tables and records together. In an object-oriented database, data relationships are defined and managed using inheritance and polymorphism, which allow objects to take on different forms or behaviors depending on the context in which they are used.

3. **Query language:** In a relational database, data is queried and manipulated using a structured query language (SQL), which is a standardized and widely used language for managing and querying data. In an object-oriented database, data is queried and manipulated using object-oriented query languages, which are specialized languages that are designed for managing and querying object-oriented data.

4. **Data manipulation:** In a relational database, data manipulation is typically performed using SQL queries, which can be used to insert, update, delete, and retrieve data from the database. In an object-oriented database, data manipulation is performed using the operations or methods defined on the objects themselves, which allows for the efficient manipulation of complex data structures and relationships.

Parallel databases

Parallel databases are designed to efficiently process large volumes of data by distributing the workload across multiple computing resources. Here are some key properties of parallel databases:

1. Data Partitioning: Parallel databases divide the data into smaller subsets called partitions. Each partition is stored on a separate computing node, allowing for parallel processing. Data partitioning can be done based on different criteria, such as range partitioning or hash partitioning.

2. Parallel Query Execution: Parallel databases are capable of executing queries simultaneously across multiple partitions or computing nodes. This parallelism enables faster data retrieval and processing, as the workload is distributed among multiple resources.

3. Load Balancing: Parallel databases employ load balancing techniques to evenly distribute the query workload across all available computing nodes. This ensures that each node receives a balanced amount of work, optimizing the overall system performance.

4. Interconnectivity: Parallel databases rely on high-speed interconnects to facilitate efficient communication and data transfer between computing nodes. This minimizes the latency associated with inter-node communication and enables fast data exchange during query execution.

5. Fault Tolerance: Parallel databases often incorporate fault-tolerant mechanisms to ensure system reliability. Redundancy techniques such as data replication and backup enable data recovery in case of node failures or system crashes, ensuring uninterrupted operation.

6. Scalability: Parallel databases offer scalability by allowing the addition of more computing nodes to the system as the data volume or processing requirements increase. This horizontal scaling approach ensures that the database can handle larger workloads by distributing them across additional resources.

7. Shared-Nothing Architecture: Parallel databases typically adopt a shared-nothing architecture, where each computing node operates independently and has its own

dedicated storage and processing capabilities. This architecture minimizes contention and allows for efficient parallel execution.

8. Parallel Indexing: To support fast data retrieval, parallel databases often employ parallel indexing techniques. Indexes are distributed across multiple nodes, enabling concurrent index scans and faster query processing.

9. Data Consistency and Synchronization: Parallel databases employ mechanisms to ensure data consistency and synchronization across multiple nodes. Techniques such as distributed transactions and distributed locking protocols are used to maintain data integrity in a parallel processing environment.

10. Query Optimization: Parallel databases employ sophisticated query optimization techniques to determine the most efficient execution plans for queries. These optimizations take into account factors such as data distribution, partitioning strategy, and resource availability to minimize execution time and resource utilization.

Overall, parallel databases provide high-performance data processing capabilities by leveraging parallelism, partitioning, load balancing, and fault tolerance techniques. These properties enable efficient handling of large datasets and support the growing demands of modern data-intensive applications.

Database Security

Database security includes a variety of measures used to secure database management systems from malicious cyber-attacks and illegitimate use. Database security programs are designed to protect not only the data within the database, but also the data management system itself, and every application that accesses it, from misuse, damage, and intrusion.

Database security encompasses tools, processes, and methodologies which establish security inside a database environment.

The major database security threats are:

1. SQL Injection Attacks

SQL injection is the most common threat. This attack is performed by entering a query into a SQL form, and if the database interprets the result as “true” it enables access to the database. These attacks usually target relational database management systems (RDBMS) based on the SQL programming language.

Databases not based on SQL (NoSQL) are not susceptible to such attacks. Instead, NoSQL databases are targeted by queries delivered by an end-user that uses commands to execute malware.

Both methods are equally threatening, getting around verification systems by obtaining credentials and then exposing the structure and content of the database. A successful attack would give an attacker free reign of everything contained within the database.

2. Malware

Malware is designed to target vulnerabilities on a network, granting access to a database, or causing damage to it. These vulnerabilities relate to unprotected endpoints on a network that can be exploited via a range of different attacks.

For IT teams to protect against malware attacks, it is important to identify the attack surface of a network. The attack surface refers to the number of vulnerabilities on a network that a cybercriminal could target.

3. Denial of Service (DoS/DDoS) Attacks

A Denial of Service (DoS) attack occurs when a database server receives more requests than it can process, causing the system to become unstable or crash. These erroneous requests can be created by an attacker and directed at a specific target. The volume of fake requests overwhelms the system, resulting in downtime for the victim.

A Distributed Denial of Service (DDoS) attack uses a botnet (a very large network of computers) to create a huge amount of traffic that even the most advanced security systems would struggle to prevent. The best defense against these types of attacks is to employ a cloud-based DoS protection service that can help to limit high and suspicious traffic volume.

4. Poor Permission Management

Many organizations fail to change the default security settings from when a database server is initially installed. Just a few years ago, as many as 20% of companies were not even changing default passwords on privileged accounts. This leaves them vulnerable to an attack from attackers who know the defaults and, more importantly, how they can be exploited.

Criminals may obtain log-in details of privileged accounts when accessing the database. Inactive accounts can also present a risk if an attacker is aware of their existence. This is why permission management should be at the forefront when developing the cybersecurity portion for your business as a whole, using zero trust protocols to prevent unauthorized access.

Occasionally, a user can be accidentally given permissions to the database that they shouldn't have access to. This presents an opportunity for hackers to target such users with phishing scams or other tactics that attempt to launch malware on their devices.

Cybercriminals can also attempt to seize control of the organization's data management system, altering privileges so they can gain database access at any time.

5. Database Backup Exposures

Backing up a database regularly is obviously recommended, but often, many of these backups are left unprotected, making them a common target for attackers. Securing backups is especially vital for industries that hold vital customer information, such as healthcare providers or banks and financial institutions.

To prevent database exposures, you should:

- Encrypt your database and any backups that are made.
- Conduct regular audits of databases and their backups to record who has been accessing this data.

6. Inadequate Auditing

Poor auditing can present a golden opportunity to cybercriminals, rendering your database non-compliant with data security regulations. Organizations are required to register all events that take place on a database server and conduct regular auditing. Of course, such auditing is best using automated systems.

A failure to implement effective auditing procedures increases the chances of a successful cyberattack. However, it is also important that any automated auditing software does not impact the overall performance of the database.

7. Unprotected Databases Due to Misconfiguration

Attacks resulting from misconfiguration are also commonly caused because of unprotected databases when some parameters and accounts are left unchanged from their initial default settings. Using these defaults, an experienced attacker can gain access. This is why businesses should always ensure their databases are being

managed correctly, using thorough procedures and audits. Database management should be conducted by an expert, whether this is an in-house professional or an external cybersecurity firm.

8. Credentials

Social engineering attacks, such as phishing or click-bait advertising can be used to obtain log-in credentials that an attacker can use to access a network and database.

9. Unencrypted data

Data encryption is a fundamental and crucial component of any cybersecurity policy, and especially when it comes to the protection of financial information. All account and financial data that is stored within your financial institution should be encrypted. This way, even if any of the data is stolen, encryption guarantees that it is unusable. In fact, at least one cybersecurity law prescribes data encryption for compliance with the regulation

Data Access Control

Database access control is a method of allowing access to company's sensitive data only to those people (database users) who are allowed to access such data and to restrict access to unauthorized persons. It includes two main components: authentication and authorization.

Authentication is a method of verifying the identity of a person who is accessing your database. Note that authentication isn't enough to protect data. An additional layer of security is required, authorization, which determines whether a user should be allowed to access the data or make the transaction he's attempting. Without authentication and authorization, there is no data security.

Any company whose employees connect to the Internet, thus, every company today, needs some level of access control implemented.

Types of Access Control

Obsolete access models include Discretionary Access Control (DAC) and Mandatory Access Control (MAC). Role Based Access Control (RBAC) is the most common method today, and the most recent model is Attribute Based Access Control (ABAC).

Discretionary Access Control (DAC)
With DAC models, the data owner allows access. DAC is a means of assigning access rights based on user-specified rules.

Mandatory Access Control (MAC)
MAC was developed using a nondiscretionary model, in which people are granted access based on an information clearance. MAC is a policy in which access rights are assigned based on central authority regulations.

Role Based Access Control (RBAC)
RBAC grants access based on a user's role and implements key security principles such as "least privilege" and "separation of privilege." Thus, someone attempting to access information can only access data necessary for their role.

Attribute Based Access Control (ABAC)
In ABAC, each resource and user are assigned a series of attributes. In this dynamic method, a comparative assessment of the user's attributes, including time of day, position and location, are used to make a decision on access to a resource.

Privileges

The authority or permission to access a named object as advised manner, for example, permission to access a table. Privileges can permit a particular user to connect to the database. In other words, privileges are the allowance to the database by the database object.

- **Database privileges**

  A privilege is permission to execute one particular type of SQL statement or access a second persons' object. Database privilege controls the use of computing resources. Database privilege does not apply to the Database administrator of the database.

- **System privileges**
  A system privilege is the right to perform an activity on a specific type of object. For example, the privilege to delete rows of any table in a database is system

privilege. There are a total of 60 different system privileges. System privileges allow users to CREATE, ALTER, or DROP the database objects.

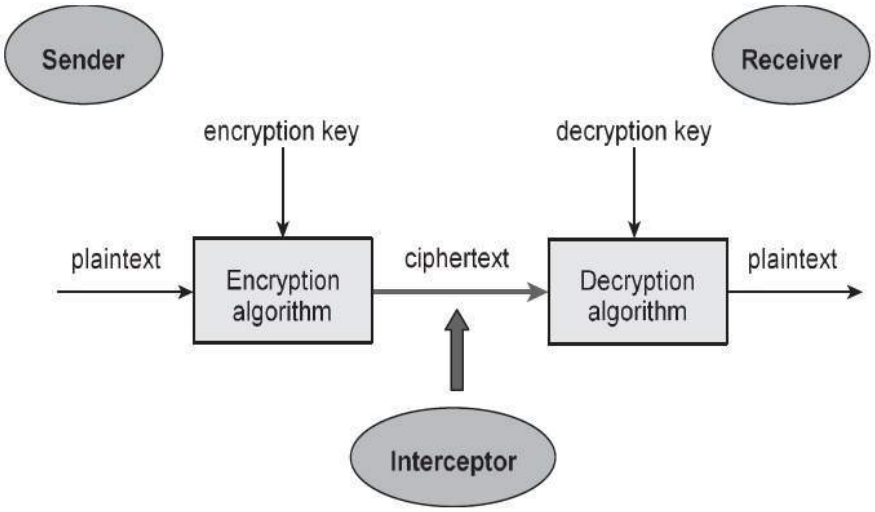
- **Object privilege**
  An object privilege is a privilege to perform a specific action on a particular table, function, or package. For example, the right to delete rows from a table is an object privilege. For example, let us consider a row of table GEEKSFORGEEKS that contains the name of the employee who is no longer a part of the organization, then deleting that row is considered as an object privilege. Object privilege allows the user to INSERT, DELETE, UPDATE, or SELECT the data in the database object.

Cryptography

Cryptography is the technique of securing information and communications through use of codes so that only those persons for whom the information is intended can understand it and process it. Thus preventing unauthorized access to information. The prefix “crypt” means “hidden” and suffix “graphy” means “writing”.

In Cryptography the techniques which are used to protect information are obtained from mathematical concepts and a set of rule based calculations known as algorithms to convert messages in ways that make it hard to decode it.

These algorithms are used for cryptographic key generation, digital signing, verification to protect data privacy, web browsing on the internet and to protect confidential transactions such as credit card and debit card transactions.



Features Of Cryptography are as follows:

Confidentiality: Information can only be accessed by the person for whom it is intended and no other person except him can access it.

Integrity: Information cannot be modified in storage or transition between sender and intended receiver without any addition to information being detected.

Non-repudiation: The creator/sender of information cannot deny his intention to send information at a later stage.

Authentication: The identities of sender and receiver are confirmed. As well as the destination/origin of information is confirmed.

Types Of Cryptography: In general there are three types Of cryptography:

Symmetric Key Cryptography: It is an encryption system where the sender and receiver of a message use a single common key to encrypt and decrypt messages. Symmetric Key Systems are faster and simpler but the problem is that sender and receiver have to somehow exchange keys in a secure manner. The most popular symmetric key cryptography systems are Data Encryption System(DES) and Advanced Encryption System(AES).

Hash Functions: There is no usage of any key in this algorithm. A hash value with fixed length is calculated as per the plain text which makes it impossible for contents of plain text to be recovered. Many operating systems use hash functions to encrypt passwords.

Asymmetric Key Cryptography: Under this system a pair of keys is used to encrypt and decrypt information. A receiver's public key is used for encryption and a receiver's private key is used for decryption. Public keys and Private keys are different. Even if the public key is known by everyone, the intended receiver can only decode it because he alone knows his private key. The most popular asymmetric key cryptography algorithm is RSA algorithm.

Statistical Databases

A statistical database (SDB) system is a database system that enables its users to retrieve only aggregate statistics (e.g., sample mean and count) for a subset of the entities represented in the database.

Statistical databases are used mainly to produce statistics about various populations. The database may contain confidential data about individuals; this information should be protected from user access. However, users are permitted to retrieve statistical information about the populations, such as averages. sums, counts, maximums, minimums, and standard deviations.

A population is a set of tuples of a relation (table) that satisfy some selection condition.

Statistical queries involve applying statistical functions to a population of tuples. For example, we may want to retrieve the number of individuals in a population or the average income in the population. However, statistical users are not allowed to retrieve individual data, such as the income of a specific person

Relevance Ranking

---

* Using Terms
  - TF-IDF
  - Similarity Based

Using Hyperlinks
  - Popularity Ranking
  - PageRank

* Using Synonyms
    - Homonyms
    - Ontologies

Crawling and Indexing the Web

Crawling:

1. Web crawling refers to the automated process of browsing the internet to discover and retrieve web pages.

2. Search engines use web crawlers, also known as spiders or bots, to systematically navigate through websites.

3. Crawlers start from a seed set of URLs and follow hyperlinks on those pages to discover new pages to crawl.

4. Robots.txt files and meta tags like "nofollow" can be used to instruct crawlers on what to exclude or follow.

5. Crawlers typically prioritize popular or frequently updated websites and revisit them periodically.

Indexing:

1. Indexing involves analyzing and storing the content of web pages to make them searchable.

2. Crawled pages are processed to extract relevant information such as text, metadata, and links.

3. Search engines create an index, which is a structured database that allows for efficient retrieval of information.

4. Indexing algorithms analyze the textual content, page structure, and other factors to determine relevance and ranking.

5. Indexing can involve language processing techniques, stemming, tokenization, and other methods to enhance search accuracy.

XML Database

XML database is a data persistence software system used for storing the huge amount of information in XML format. It provides a secure place to store XML documents.

You can query your stored data by using XQuery, export and serialize into desired format. XML databases are usually associated with document-oriented databases.

There are two types of XML databases.

XML-enable Database
An XML-enabled database works just like a relational database. It is like an extension provided for the conversion of XML documents. In this database, data is stored in a table, in the form of rows and columns.

Native XML Database
Native XML databases are used to store large amounts of data. Instead of table format, Native XML database is based on container format. You can query data by XPath expressions.

Native XML database is preferred over XML-enable database because it is highly capable to store, maintain and query XML documents.

Let's take an example of XML database:

```
<?xml version="1.0"?>
<contact-info>
  <contact1>
    <name>Vimal Jaiswal</name>
    <company>SSSIT.org</company>
    <phone>(0120) 4256464</phone>
  </contact1>
```

```
  <contact2>
    <name>Mahesh Sharma </name>
    <company>SSSIT.org</company>
    <phone>09990449935</phone>
  </contact2>
</contact-info>
```

In the above example, a table named contacts is created and holds the contacts (contact1 and contact2). Each one contains 3 entities: name, company and phone.