SNS COLLEGE OF TECHNOLOGY,COIMBATORE-35

(AN AUTONOMOUS INSTITUTION)

DEPARTMENT OF ARTIFICIAL

INTELLIGENCE AND MACHINE LEARNING

19CST201-DATABASE MANAGEMENT SYSTEM

UNIT-V

B tree:

PHYSICAL STORAGE AND MONGODB

Topic: B-Tree Index File

A **B Tree** Index is a multilevel index.
A **B Tree** is a rooted tree satisfying the following properties :

1. All paths from the root to the leaf are equally long.
2. A node that is not a root or leaf, has between **[n / 2]** and **'n'** children.
3. A leaf node has between **[(n-1) / 2]** and **'n-1'** values.

The structures of leaf, non-leaf nodes of this tree is :

(a)

(b)

Properties of B-tree

Following are some of the properties of B-tree in DBMS:

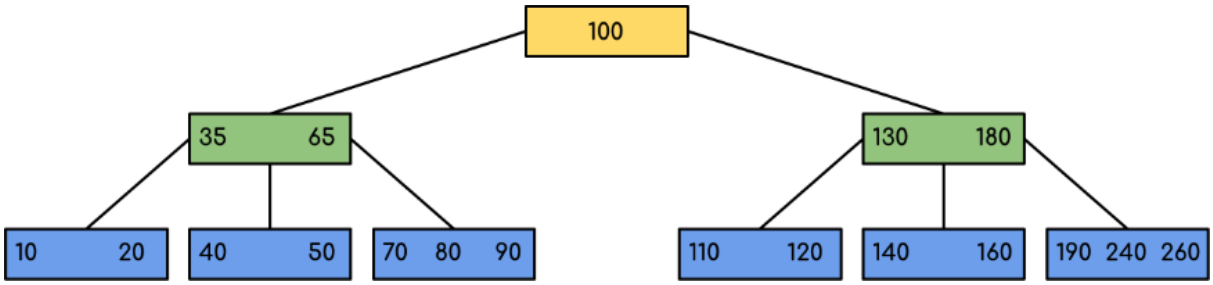
- A non-leaf node's number of keys is one less than the number of its children.
- The number of keys in the root ranges from one to (m-1) maximum. Therefore, root has a minimum of two and a maximum of m children.

- The keys range from min([m/2]-1) to max(m-1) for all nodes (non-leaf nodes) besides the root. Thus, they can have between m and [m/2] children.
- The level of each leaf node is the same.

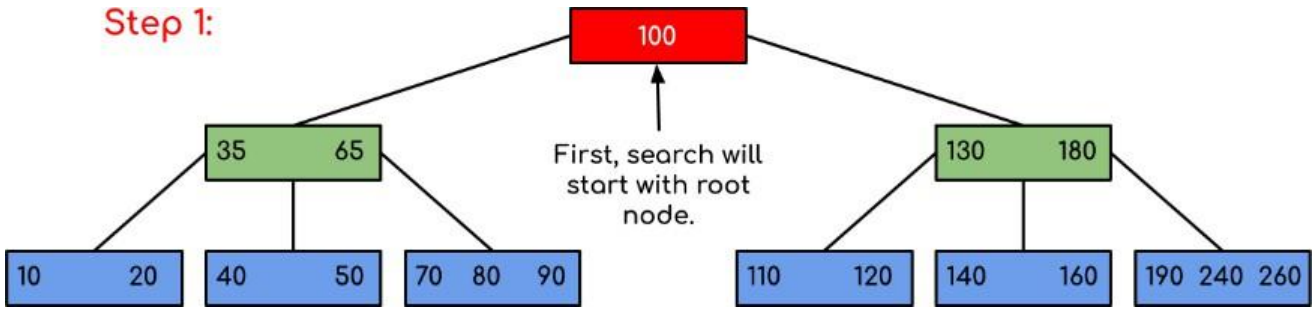
Time Complexity of B-Tree:

Sr. No.	Algorithm	Time Complexity
1.	Search	O(log n)
2.	Insert	O(log n)
3.	Delete	O(log n)

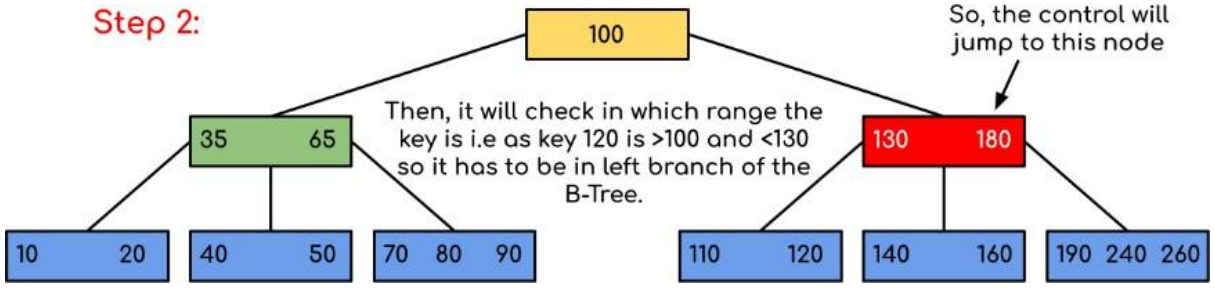
1.



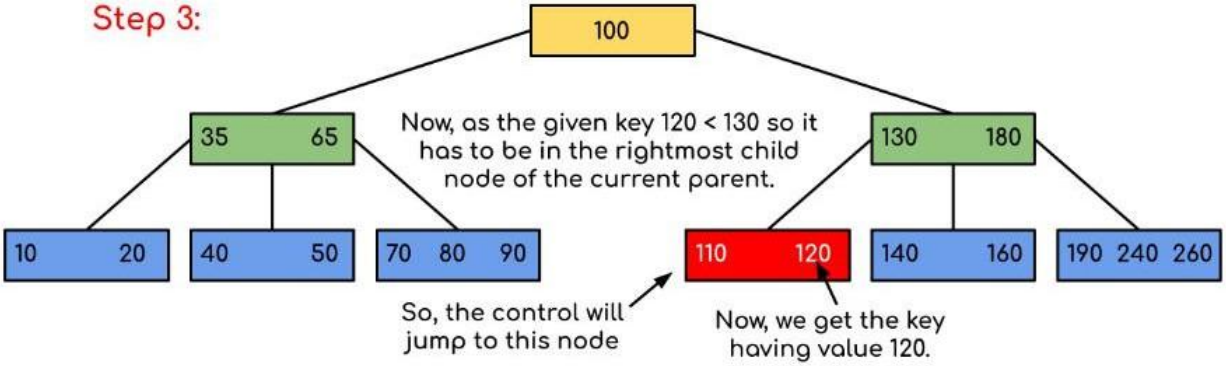
Solution:



Step 2:


100
So, the control will jump to this node
Then, it will check in which range the key is i.e as key 120 is >100 and <130 so it has to be in left branch of the B-Tree.
35 65
130 180
10 20
40 50
70 80 90
110 120
140 160
190 240 260

Step 3:


100
Now, as the given key 120 < 130 so it has to be in the rightmost child node of the current parent.
35 65
130 180
10 20
40 50
70 80 90
110 120
140 160
190 240 260
So, the control will jump to this node
Now, we get the key having value 120.

SNS COLLEGE OF TECHNOLOGY,COIMBATORE-35

(AN AUTONOMOUS INSTITUTION)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

19CST201-DATABASE MANAGEMENT SYSTEM

UNIT-V

PHYSICAL STORAGE AND MONGODB

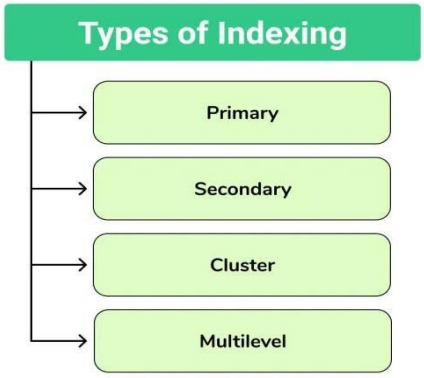
Indexes: **Topic: Indexes**

- An index takes a search key as input.
- Efficiently returns a collection of matching records.
- An Index is a small table having only two columns.
- The first column comprises a copy of the primary or candidate key of a table.
- The second column contains a set of pointers for holding the address of the disk block where that specific key-value is stored.

There are mainly 4 types of indexing methods

- Primary Indexing
- Secondary Indexing
- Cluster Indexing
- Multilevel Indexing

Types of Indexing

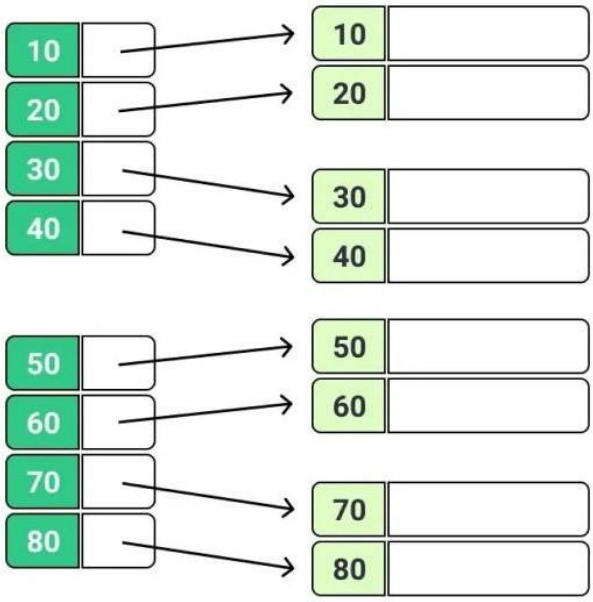


Primary Indexing

- Primary Index is an ordered file which has fixed length size with two fields.
- The first field is the same a primary key and second field is pointed to that specific data block.
- In the primary index, there is always one to one relationship between the entries in the index table.
- Primary Indexing is further divided into two types.
    - Dense Index
    - Sparse Index

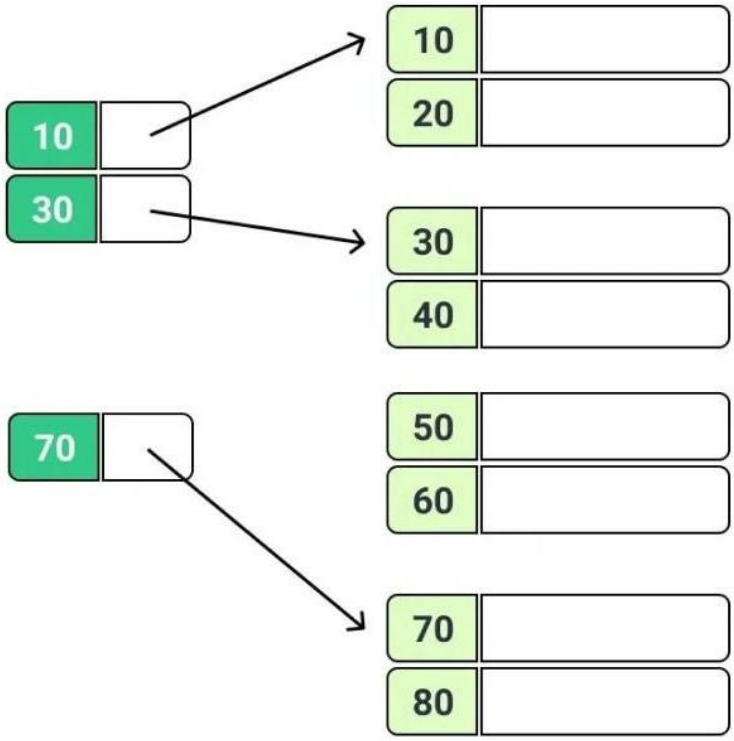
Dense Index

- In a dense index, a record is created for every search key valued in the database.
- Dense indexing helps you to search faster but needs more space to store index records.
- In dense indexing, records contain search key value and points to the real record on the disk



Sparse Index

- The sparse index is an index record that appears for only some of the values in the file.
- Sparse Index helps you to resolve the issues of dense indexing.
- In sparse indexing technique, a range of index columns stores the same data block address, and when data needs to be retrieved, this block address will be fetched.
- Sparse indexing method stores index records for only some search key values.
- It needs less space, less maintenance overhead for insertion, and deletions but it is slower compared to the dense index for locating records.

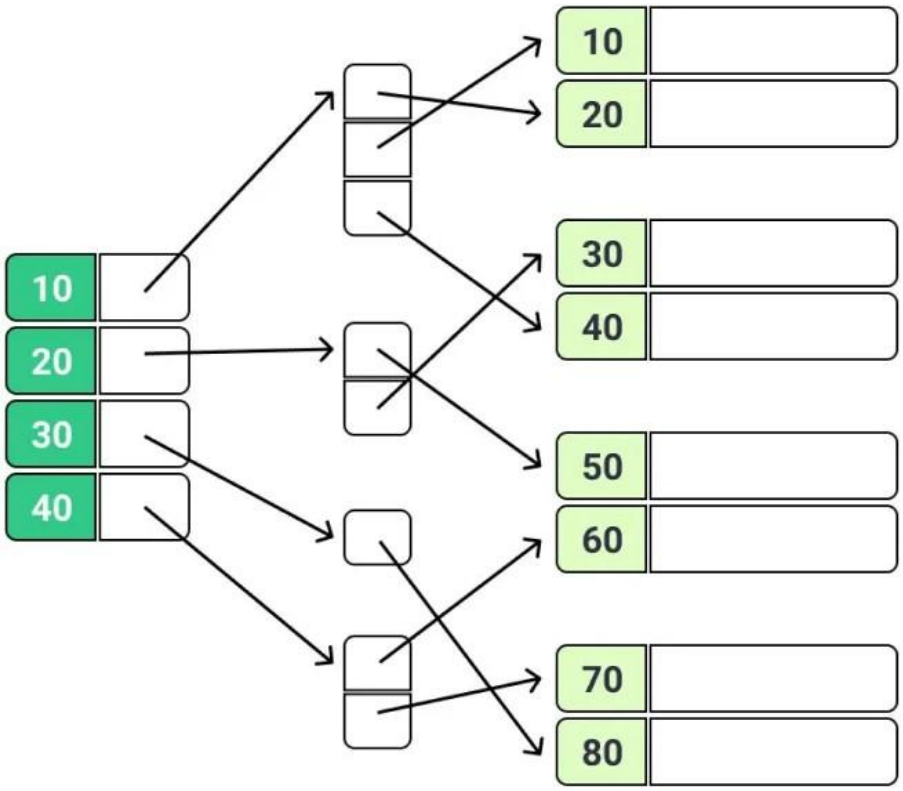


Secondary Indexing

- The secondary index can be generated by a field which has a unique value for each record.
- It is also known as a non-clustering index.
- This two-level database indexing technique is used to reduce the mapping size of the first level.
- For the first level, a large range of numbers is selected, because of this mapping size always remains small.

Example

- In a bank account database, data is stored sequentially by Account_No, we may want to find all accounts in of a specific branch of some bank.
- In this case, we can have a secondary index for every search key.
- Index record is a record pointing to a bucket that contains pointers to all the records with their specific search key value.



Cluster Indexing:

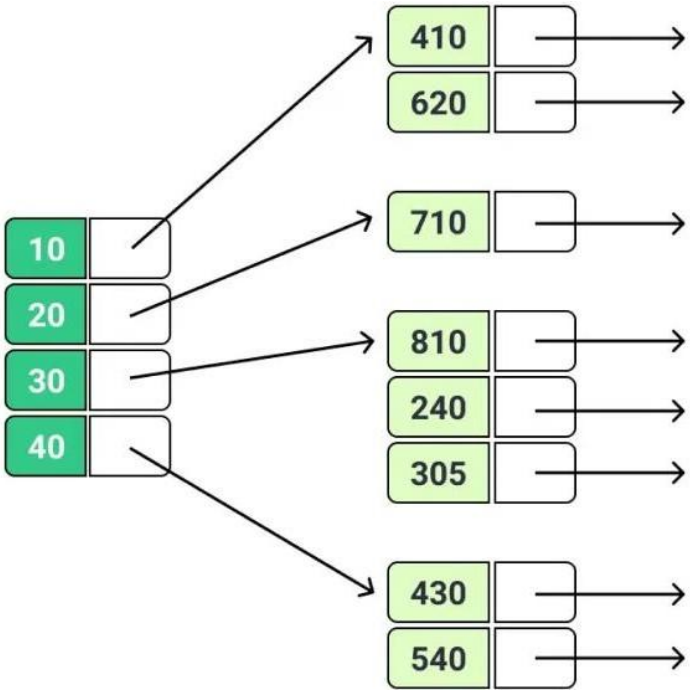
- In a clustered index, records themselves are stored in the index and not pointers.
- Sometimes the index is created on non-primary key columns which might not be unique for each record. In such a situation, you can group two or more columns to get the unique values and create an index which is called clustered Index.
- This also helps you to identify the record faster.

Example

- Consider a company recruited many employees in various departments. In this case, ***c***lustering index should be created for all employees who belong to the same dept.
- In a single cluster it is considered that an index points to the cluster as a whole.

Multilevel Indexing

- Multilevel Indexing is created when a primary index does not fit in memory.
- In this type of indexing method, you can reduce the number of disk accesses to short any record and kept on a disk as a sequential file and create a sparse base on that file.



SNS COLLEGE OF TECHNOLOGY,COIMBATORE-35

(AN AUTONOMOUS INSTITUTION)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

19CST201-DATABASE MANAGEMENT SYSTEM

UNIT-V

PHYSICAL STORAGE AND MONGODB

Topic: Ordered Indices

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 record and each of which is 10 bytes long. If their IDs start with 1, 2, 3....and so on and we have to search student 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.

SNS COLLEGE OF TECHNOLOGY,COIMBATORE-35

(AN AUTONOMOUS INSTITUTION)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

19CST201-DATABASE MANAGEMENT SYSTEM

UNIT-V

PHYSICAL STORAGE AND MONGODB

RAID:

Topic: RAID

RAID (redundant array of independent disks) is a way of storing the same data in different places on multiple hard disks or solid-state drives (SSDs) to protect data in the case of a drive failure. There are different RAID levels, however, and not all have the goal of providing redundancy.

RAID works

RAID works by placing data on multiple disks and allowing input/output (I/O) operations to overlap in a balanced way, improving performance. Because using multiple disks increases the mean time between failures, storing data redundantly also increases fault tolerance.

RAID arrays appear to the operating system (OS) as a single logical drive.

RAID employs the techniques of disk mirroring or disk striping. Mirroring will copy identical data onto more than one drive. Striping partitions help spread data over multiple disk drives. Each drive's storage space is divided into units ranging from a sector of 512 bytes up to several megabytes. The stripes of all the disks are interleaved and addressed in order. Disk mirroring and disk striping can also be combined in a RAID array.

In a single-user system where large records are stored, the stripes are typically set up to be small (512 bytes, for example) so that a single record spans all the disks and can be accessed quickly by reading all the disks at the same time.

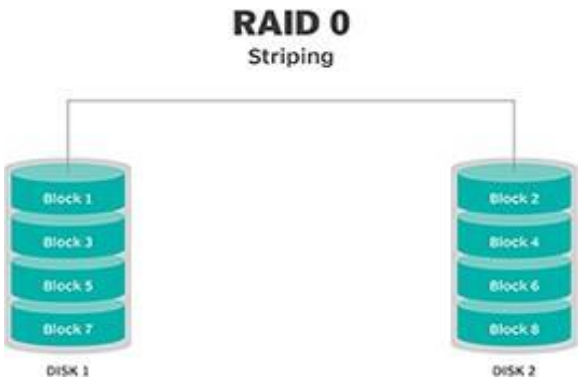
In a multiuser system, better performance requires a stripe wide enough to hold the typical or maximum size record, enabling overlapped disk I/O across drives.

RAID levels

RAID devices use different versions, called levels. The original paper that coined the term and developed the RAID setup concept defined six levels of RAID -- 0 through 5. This numbered system enabled those in IT to differentiate RAID versions. The number of levels has since expanded and has been broken into three categories: standard, nested and nonstandard RAID levels.

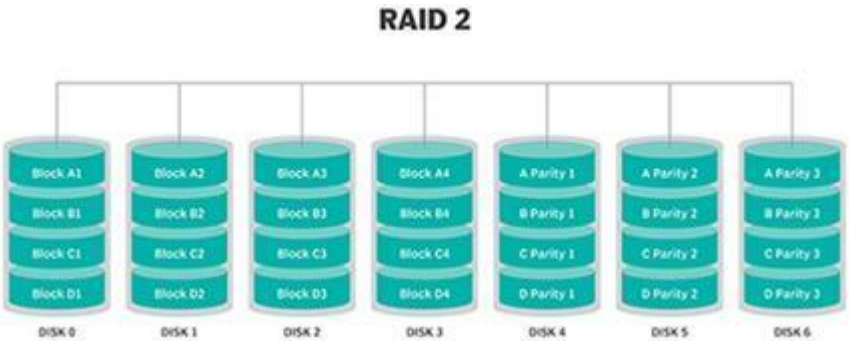
Standard RAID levels

RAID 0. This configuration has striping but no redundancy of data. It offers the best performance, but it does not provide fault tolerance.

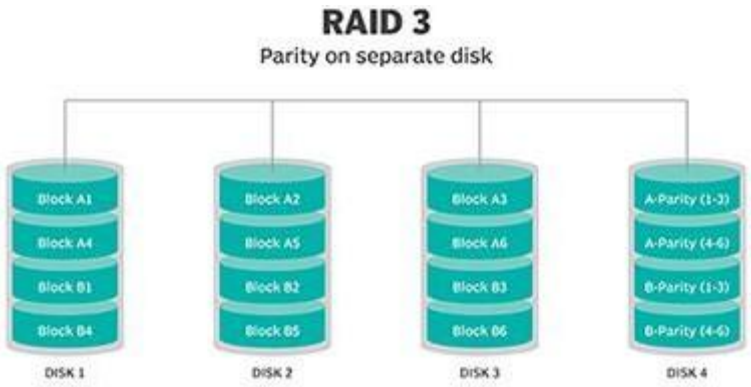


RAID 1. Also known as *disk mirroring*, this configuration consists of at least two drives that duplicate the storage of data. There is no striping. Read performance is improved, since either disk can be read at the same time. Write performance is the same as for single disk storage.

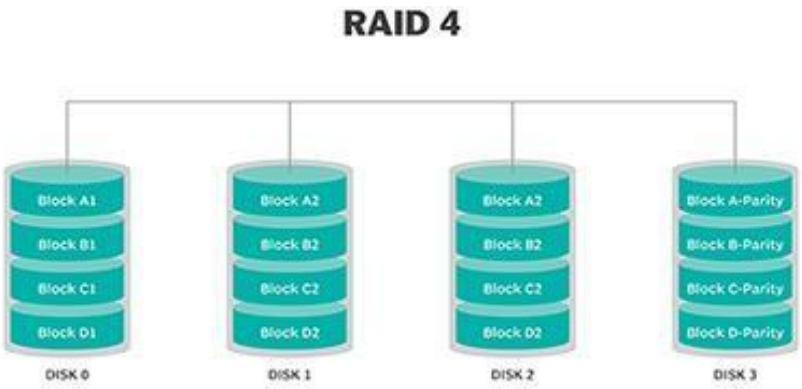
RAID 2. This configuration uses striping across disks, with some disks storing error checking and correcting (ECC) information. RAID 2 also uses a dedicated Hamming code parity, a linear form of ECC. RAID 2 has no advantage over RAID 3 and is no longer used.



RAID 3. This technique uses striping and dedicates one drive to storing parity information. The embedded ECC information is used to detect errors. Data recovery is accomplished by calculating the exclusive information recorded on the other drives. Because an I/O operation addresses all the drives at the same time, RAID 3 cannot overlap I/O. For this reason, RAID 3 is best for single-user systems with long record applications.

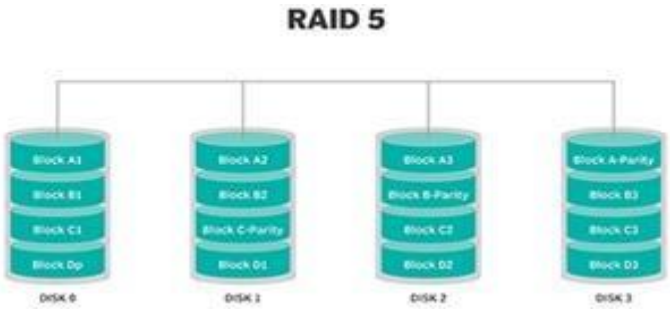


RAID 4. This level uses large stripes, which means a user can read records from any single drive. Overlapped I/O can then be used for read operations. Because all write operations are required to update the parity drive, no I/O overlapping is possible.

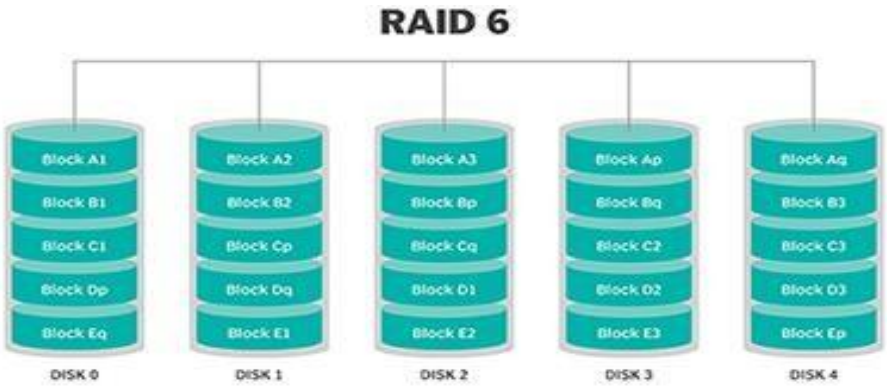


RAID 5. This level is based on parity block-level striping. The parity information is striped across each drive, enabling the array to function, even if one drive were to fail. The array's architecture enables read and write operations to span multiple drives. This results in performance better than that of a single drive, but not as high as a RAID 0 array. RAID 5 requires at least three disks, but it is often recommended to use at least five disks for performance reasons.

RAID 5 arrays are generally considered to be a poor choice for use on write-intensive systems because of the performance impact associated with writing parity data. When a disk fails, it can take a long time to rebuild a RAID 5 array.



RAID 6. This technique is similar to RAID 5, but it includes a second parity scheme distributed across the drives in the array. The use of additional parity enables the array to continue functioning, even if two disks fail simultaneously. However, this extra protection comes at a cost. RAID 6 arrays often have slower write performance than RAID 5 arrays.



SNS COLLEGE OF TECHNOLOGY,COIMBATORE-35

(AN AUTONOMOUS INSTITUTION)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

19CST201-DATABASE MANAGEMENT SYSTEM

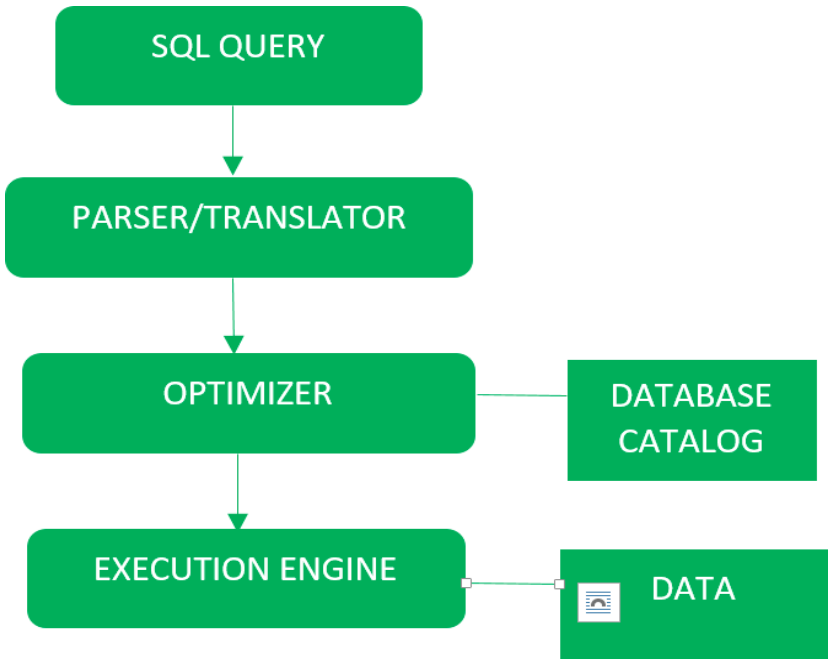
UNIT-V

PHYSICAL STORAGE AND MONGODB

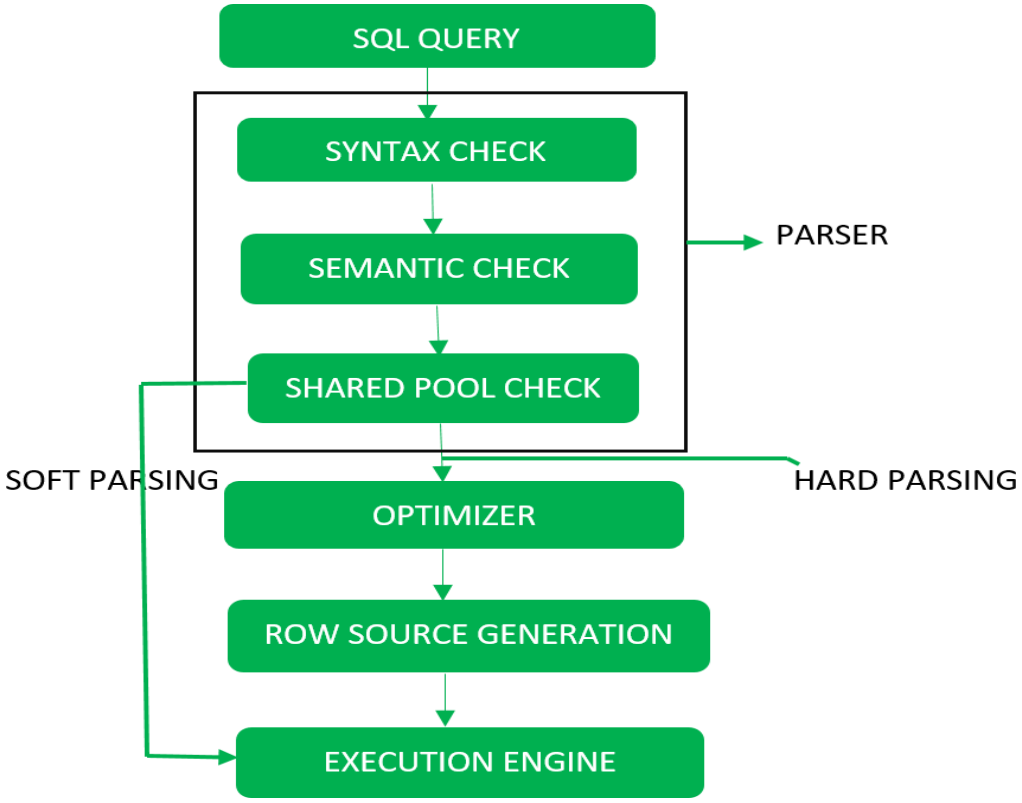
Topic: Query Processing Overview

Query Processing Overview:

Query Processing includes translations on high level Queries into low level expressions that can be used at physical level of file system, query optimization and actual execution of query to get the actual result.
Block Diagram of Query Processing is as:



Detailed Diagram is drawn as:



It is done in the following steps:

- **Step-1:**

  Parser: During parse call, the database performs the following checks- Syntax check, Semantic check and Shared pool check, after converting the query into relational algebra.
  Parser performs the following checks as (refer detailed diagram):

  1. **Syntax check –** concludes SQL syntactic validity. Example:
     SELECT * FORM employee

     Here error of wrong spelling of FROM is given by this check.
  2. **Semantic check –** determines whether the statement is meaningful or not. Example: query contains a tablename which does not exist is checked by this check.
  3. **Shared Pool check –** Every query possess a hash code during its execution. So, this check determines existence of written hash code in shared pool if code exists in shared pool then database will not take additional steps for optimization and execution.

Hard Parse and Soft Parse –
If there is a fresh query and its hash code does not exist in shared pool then that query has to pass through from the additional steps known as hard parsing otherwise if hash code exists then query does not passes through additional steps. It just passes directly to execution engine (refer detailed diagram). This is known as soft parsing.
Hard Parse includes following steps – Optimizer and Row source generation.

- **Step-2:**

  Optimizer: During optimization stage, database must perform a hard parse atleast for one unique DML statement and perform optimization during this parse. This database never optimizes DDL unless it includes a DML component such as subquery that require optimization.
  It is a process in which multiple query execution plan for satisfying a query are examined and most efficient query plan is satisfied for execution. Database catalog stores the execution plans and then optimizer passes the lowest cost plan for execution.

  Row Source Generation –
  The Row Source Generation is a software that receives a optimal execution plan from the optimizer and produces an iterative execution plan that is usable by the rest of the database. the iterative plan is the binary program that when executes by the sql engine produces the result set.

- **Step-3:**

  Execution Engine: Finally runs the query and display the required result.