VELAMMAL COLLEGE OF ENGINEERING AND TECHNOLOGY, MADURAI – 625 009

Department of Computer Science and Engineering

INTERNAL ASSESMENT TEST – III-KEY

Branch	CSE& IT	**Year/Sem./Sec.**	II/IV/A&B
Course Code	21CS206	**Date**	09-05-2023
Course Name	Database Management Systems	**Max. marks**	100
Course Incharge	Dr.A.M Rajeswari	**Time**	03:00 Hours

PART A (Answer All Questions) (8 X 2 = 16)

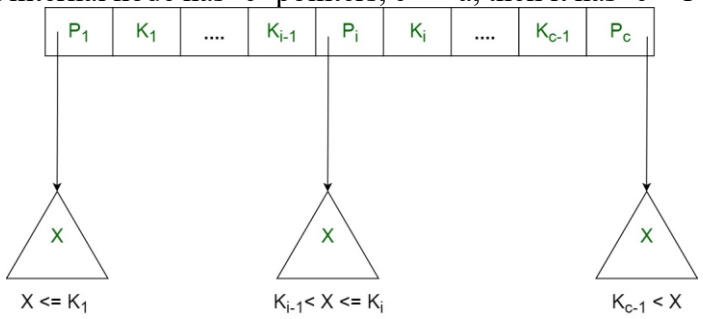
		K	CO	Marks
1.	**What is the goal of recovery systems in DBMS?** An integral part of a database system is a recovery scheme that can restore the database to the consistent state that existed before the failure.	K1	CO4	2
2.	**List the types of storage structure that take part in the recovery Systems.** • When the system recovers from a failure, it can restore the latest dump. • It can maintain a redo-list and an undo-list as checkpoints. • It can recover the system by consulting undo-redo lists to restore the state of all transactions up to the last checkpoint.	K1	CO4	2
3.	**Mention the use of check points in the case of Log based recovery systems.** • **Transaction identifier:** Unique Identifier of the transaction that performed the write operation. • **Data item:** Unique identifier of the data item written. • **Old value:** Value of data item prior to write. • **New value:** Value of data item after write operation.	K1	CO4	2
4.	**List out the steps involved in query processing.** Query processing refers to the range of activities involved in extracting data from a database. The basic steps are: • Parsing and translation • optimization • evaluation	K1	CO5	2
5.	**Give the parameters to be considered for cost based optimization.** – Access cost to secondary storage. – Storage cost. – Computation cost. – Memory uses cost. – Communication cost	K1	CO5	2

6.	**Compare Heap file organization and Sorted file Organization** A heap file has good storage efficiency and supports fast scanning and insertion of records. However, it is slow for searches and deletions. A sorted file also offers good storage efficiency, but insertion and deletion of records is slow. Searches are faster than in heap files.	K1	CO5	2
7.	**When will you use multi level index?** If primary index does not fit in memory, access becomes expensive. To reduce number of disk accesses to index records, treat primary index kept on disk as a sequential file and construct a sparse index on it. outer index – a sparse index of primary index inner index – the primary index file If even outer index is too large to fit in main memory, yet another level of index can be created, and so on	K1	CO5	2
8.	**How a new record can be inserted into the sequential file structure, when there is no space to accommodate any new record?** In a sequential file structure, where records are stored in a contiguous block of memory, inserting a new record when there is no space to accommodate it requires reorganizing the file to make space for the new record. This can be done using one of the following techniques: 1. Overflow area: A separate overflow area can be created to store new records when there is no space in the main file. This area can be reserved for new records and managed separately from the main file. When a new record is inserted, it is written to the overflow area, and a pointer to its location is stored in the main file. This technique is simple to implement but can lead to slower access times as records are scattered across different locations. 2. Compaction: The file can be reorganized by moving records around to create a contiguous block of free space for the new record. This can be done by shifting all the records after the insertion point by a certain amount to create a gap for the new record. This technique requires more processing overhead but can lead to faster access times as records are stored in a contiguous block. 3. Extend the file: If the file system allows, the size of the file can be increased to make space for new records. This can be done by adding more blocks of memory to the file. However, this technique has limits as file size is often limited by the operating system or storage device.	K1	CO5	2
9.	**List out the reasons why the records of the same relation vary in size.** The records of the same relation may vary in size due to the following reasons: 1. Variable length fields: Some fields in a record may have variable lengths, such as text fields, which can store a varying number of characters. The length of these fields can vary from record to record, leading to variations in record size. 2. Nullable fields: Some fields in a record may be allowed to have null values, indicating that the field has no value. These fields take up less	K1	CO5	2

	space in records where they have null values, resulting in smaller records. 3. Field data types: Different data types have different storage requirements. For example, a field storing an integer takes up less space than a field storing a string of the same length. If records have fields with different data types, their sizes will vary accordingly. 4. Record format: The format of a record can also affect its size. For example, if a record has a header that contains metadata about the record, such as a record ID or timestamp, this will increase the size of the record. 5. Number of fields: The number of fields in a record can also affect its size. Records with more fields take up more space than records with fewer fields, all other things being equal.			
10.	**Compare and contrast Static hashing and Dynamic hashing techniques.** Static hashing is a hashing technique that allows users to perform lookups on a finalized dictionary set (all objects in the dictionary are final and not changing). In contrast, dynamic hashing is a hashing technique in which the data buckets are added and removed dynamically and on demand	K2	CO5	2
11.	**Differentiate between the log records <Ti, X, V1, V2 > and <Ti, X, V >. Justify your answer with a proper example.** Both <Ti, X, V1, V2> and <Ti, X, V> are log records used in database management systems to keep track of changes to data items. However, they differ in the amount of information they record. The log record <Ti, X, V1, V2> records both the old value (V1) and new value (V2) of a data item (X) that has been modified by a transaction (Ti). This type of log record is used when the database system needs to support undo and redo operations, such as during database recovery after a system failure. For example, consider a banking application where a transaction T1 transfers $100 from account A to account B. The log record for this transaction could be <T1, A, 500, 400> to indicate that the old value of A was 500 and the new value is 400 after the transfer. On the other hand, the log record <Ti, X, V> records only the new value (V) of a data item (X) that has been modified by a transaction (Ti). This type of log record is used when the database system does not need to support undo and redo operations, such as in a read-only database.	K2	CO5	2

	For example, consider a book rental system where a transaction T2 updates the rental status of a book to 'checked out'. The log record for this transaction could be <T2, Book_123, checked out> to indicate that the new value of the rental status for Book_123 is 'checked out'. In summary, the log record <Ti, X, V1, V2> records both the old and new values of a data item modified by a transaction and is used for undo and redo operations, while the log record <Ti, X, V> records only the new value of a data item and is used in read-only databases.			
12.	**Summarize B tree and B+ tree.** A B+-Tree index takes the form of a balanced tree in which every path from the root of the root of the root of the tree to a leaf of the tree is of the same length. A B-tree eliminates the redundant storage of search-key values .It allows search key values to appear only once	K2	CO5	2
13.	**What is Query Optimization?** Query optimization refers to the process of finding the lowest cost method of evaluating a given query.	K1	CO5	2
14.	**List the parameters to be considered for query cost estimation.** • Number of disk accesses. • Execution time taken by the CPU to execute a query. • Communication costs in distributed or parallel database systems.	K1	CO5	2
15.	**Mention any four of the algorithms used in optimizing the SELECT operation.** • Linear Search, • Primary B^+-tree index, • Equality on Key, • Secondary B^+-tree index, • Equality on Key, • Secondary B^+-tree index, • Comparison	K1	CO5	2
16.	**Give the relational expression tree for the query "SELECT emp_id, emp_name, project_name FROM employee, project WHERE project_location = 'MADURAI';** 	K2	CO5	2

17.	**The set operations union and intersection are commutative. Demonstrate with an example.** The union and intersection of sets may be seen as analogous to the addition and multiplication of numbers. Like addition and multiplication, the operations of union and intersection are commutative and associative, and intersection distributes over union.	K2	CO5	2
18.	**Explain 'Heuristic Optimization' of query processing with an example.** Perform SELECT and PROJECT operations as early as possible to reduce the number of tuples and attributes The SELECT and JOIN operations that are most restrictive should be executed before other similar operations	K2	CO5	2
19.	**Define the term NoSQL.** NoSQL, also referred to as "not only SQL", "non-SQL", is an approach to database design that enables the storage and querying of data outside the traditional structures found in relational databases.	K1	CO6	2
20.	**Why NoSQL architecture is called as shared nothing architecture?** In shared nothing architecture, data is partitioned in some manner and spread across a set of machines. This means that each machine has sole access, and hence sole responsibility, for the data it holds. It does not share responsibility with other machines. So data is completely segregated, with each node having total autonomy over its particular subset. As data is partioned and distributed in NoSQL it similar to shared nothing architecture.	K1	CO6	2
21.	**Define CAP theorem**. The CAP theorem **maintains that a distributed system can deliver only two of three desired characteristics: consistency, availability, and partition tolerance**.	K1	CO6	2
22.	**Differentiate Scale-Up and Scale-Out in NoSQL.** Scaling up (or vertical scaling) is adding more resources—like CPU, memory, and disk—to increase more compute power and storage capacity. This term applies to traditional applications deployed on physical servers or virtual machines as well. Scaling out (or horizontal scaling) addresses some of the limitations of the scale up method. With horizontal scaling, the compute resource limitations from physical hardware are no longer the issue. In fact, you can use any reasonable size of server as long as the server has enough resources to run the pods.	K2	CO6	2
23.	**Compare RDBMS and NoSQL.** NoSql database implementation is easy and typically uses cheap servers to manage the exploding data and transaction while RDBMS databases are expensive and it uses big servers and storage systems. So the storing and processing data cost per gigabyte in the case of NoSQL can be many times lesser than the cost of RDBMS.	K2	CO6	2

24.	**Illustrate the 'Key-Value' pair with an example.** A key-value database (also known as a key-value store) is a type of noSQL database. Unlike prior relational databases that stored data in defined tables and columns, a key-value database instead uses individual or combinations of keys to retrieve associated values. Together they are known as key-value pairs.	K2	CO6	2
	PART B (Answer All Questions) (2 X 13 = 26)			
1.	**Demonstrate the construction of B+ tree with a suitable example.** **<u>B+ Trees</u>** B + tree is a variation of B-tree data structure. In a B + tree, data pointers are stored only at the leaf nodes of the tree. In a B+ tree structure of a leaf node differs from the structure of internal nodes. The leaf nodes have an entry for every value of the search field, along with a data pointer to the record (or to the block that contains this record). The leaf nodes of the B+ tree are linked together to provide ordered access on the search field to the records. **The structure of the internal nodes of a B+ tree of order 'a' is as follows:** 1. Each internal node is of the form: <P_1, K_1, P_2, K_2, ….., P_{c-1}, K_{c-1}, P_c> where c <= a and each P_i is a tree pointer (i.e points to another node of the tree) and, each K_i is a key-value (see diagram-I for reference). 2. Every internal node has : $K_1 < K_2 < …. < K_{c-1}$ 3. For each search field values 'X' in the sub-tree pointed at by P_i, the following condition holds : $K_{i-1} < X <= K_i$, for $1 < i < c$ and, $K_{i-1} < X$, for i = c (See diagram I for reference) 4. Each internal node has at most 'a' tree pointers. 5. The root node has, at least two tree pointers, while the other internal nodes have at least \ceil(a/2) tree pointers each. 6. If an internal node has 'c' pointers, c <= a, then it has 'c – 1' key values.  **Diagram-I The structure of the leaf nodes of a B+ tree of order 'b' is as follows:** 1. Each leaf node is of the form: <<K_1, D_1>, <K_2, D_2>, ….., <K_{c-1}, D_{c-1}>, P_{next}> where c <= b and each D_i is a data pointer (i.e points to actual record in the disk whose key value is K_i or to a disk file block containing that record) and, each K_i is a key value and, P_{next} points to next leaf node in the B+ tree (see diagram II for reference). 2. Every leaf node has : $K_1 < K_2 < …. < K_{c-1}$, c <= b 3. Each leaf node has at least \ceil(b/2) values. 4. All leaf nodes are at the same level.	K2	CO5	13

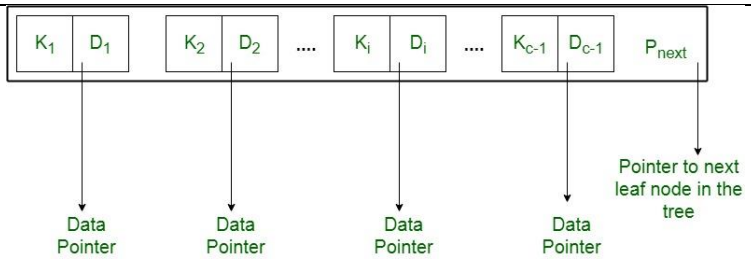
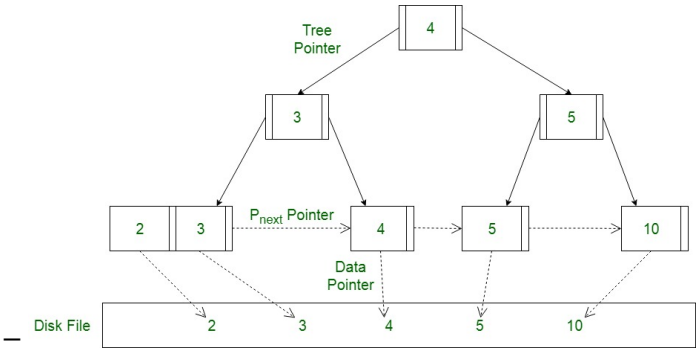


Diagram-II Using the P_{next} pointer it is viable to traverse all the leaf nodes, just like a linked list, thereby achieving ordered access to the records stored in the disk. A Diagram of B+ Tree



Advantages of B+Trees:

- A B+ tree with 'l' levels can store more entries in its internal nodes compared to a B-tree having the same 'l' levels. This accentuates the significant improvement made to the search time for any given key. Having lesser levels and the presence of P_{next} pointers imply that the B+ trees is very quick and efficient in accessing records from disks.
- Data stored in a B+ tree can be accessed both sequentially and directly.
- It takes an equal number of disk accesses to fetch records.
- B+trees have redundant search keys, and storing search keys repeatedly is not possible.

Disadvantages of B+Trees:

- The major drawback of B-tree is the difficulty of traversing the keys sequentially. The B+ tree retains the rapid random access property of the B-tree while also allowing rapid sequential access.

Application of B+ Trees:

- Multilevel Indexing
- Faster operations on the tree (insertion, deletion, search)
- Database indexing

2.	**Demonstrate the use of extendable hashing technique with a suitable example. Assume the bucket size as 2.** use of extendable hashing technique with an example where the bucket size is 2. Initially, the hash table consists of a single bucket: **[B0]** where B0 is the bucket. Now, let's insert the following keys into the hash table:	K2	CO5	13

10
20
30
40
50
60
70
80

Since the bucket size is 2, each bucket can hold up to 2 keys. We start by inserting the key 10 into the bucket B0. The bucket now looks like this:

[10]

Next, we insert the key 20 into the bucket. The bucket now looks like this:

[10 | 20]

Now, we insert the key 30. Since the bucket is full, we need to split it into two buckets. We create a new bucket, B1, and split the keys between the two buckets based on the most significant bit of their hash value:

[0]
**| **
[10] [20]
B0 B1

We then insert the key 30 into the bucket B1, which now looks like this:

[30]

Next, we insert the key 40, which hashes to the same bucket as 20. Since the bucket is full, we split it into two buckets:

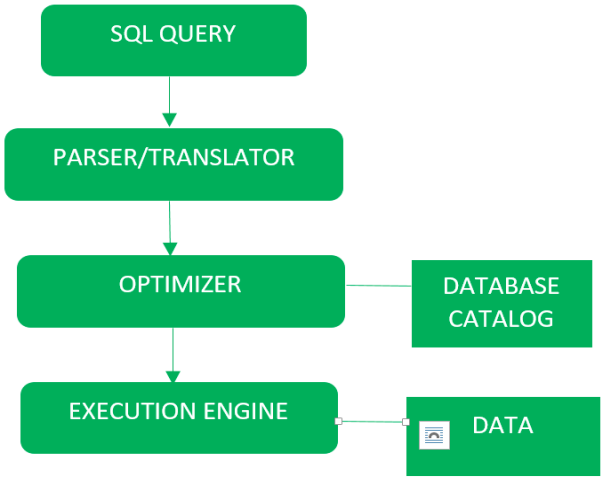
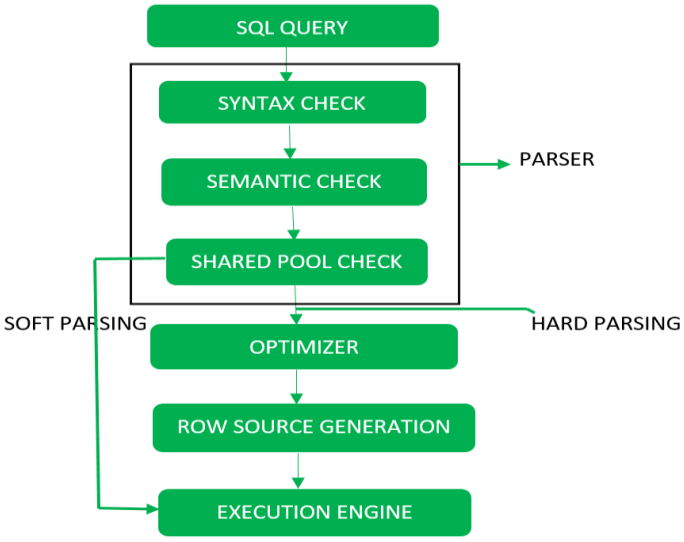
[1]
**| **
[20] [40]
B1 B2

We continue inserting keys in this manner until the hash table looks like this:

[0]
**| **
[10] [20]
B0 B1

[1]
**| **
[30] [40 | 50]
B2 B3

[1]
**| **
[60] [70 | 80]
B4 B5

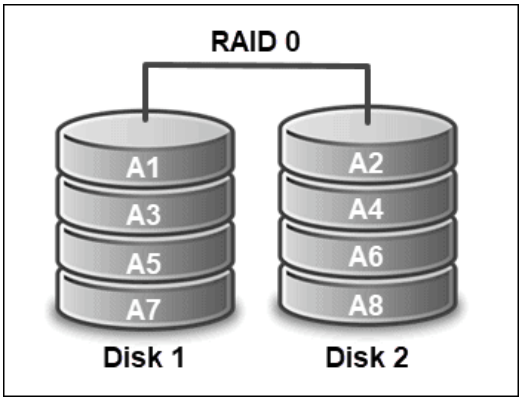
	Here, the hash table consists of 6 buckets, each of which contains 2 keys except for bucket B3, which contains 3 keys. The hash table is balanced, and all of the buckets are at the same level, making it efficient for searching and indexing data. That's how extendable hashing technique can be used to efficiently handle data insertion in a dynamic environment.			
3.	**Explain how the query processing is done in detail?** 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.	K2	CO5	13

	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.			
4.	**Explain Different types of RAID levels with example** RAID (redundant array of independent disks) is a setup consisting of multiple disks for data storage. They are linked together to prevent data loss and/or speed up performance. Having multiple disks allows the employment of various techniques like disk striping, disk mirroring, and parity. **RAID Levels and Types** RAID levels are grouped into the following categories: • Standard RAID levels • Non-standard RAID levels • Nested/hybrid RAID levels Additionally, you can choose how to implement RAID on your system. Therefore you can choose between hardware RAID, software RAID, and firmware RAID. The following list explains the standard RAID levels (0, 1, 2, 3, 4, 5, 6) and popular non-standard and hybrid options (RAID 10).	K2	CO5	13

RAID 0: Striping

RAID 0, also known as a striped set or a striped volume, requires a minimum of two disks. The disks are merged into a single large volume where data is stored evenly across the number of disks in the array.

This process is called disk striping and involves splitting data into blocks and writing it simultaneously/sequentially on multiple disks. Configuring the striped disks as a single partition increases performance since multiple disks do reading and writing operations simultaneously. Therefore, RAID 0 is generally implemented to improve speed and efficiency.



It is important to note that if an array consists of disks of different sizes, each will be limited to the smallest disk size in the setup. This means that an array composed of two disks, where one is 320 GB, and the other is 120 GB, actually has the capacity of 2 x 120 GB (or 240 GB in total).

Certain implementations allow you to utilize the remaining 200 GB for different use. Additionally, developers can implement multiple controllers (or even one per disk) to improve performance.

RAID 0 is the most affordable type of redundant disk configuration and is relatively easy to set up. Still, it does not include any redundancy, fault tolerance, or party in its composition. Hence, problems on any of the disks in the array can result in complete data loss. This is why it should only be used for non-critical storage, such as temporary files backed up somewhere else.

Advantages of RAID 0

- Cost-efficient and straightforward to implement.
- Increased read and write performance.
- No overhead (total capacity use).

Disadvantages of RAID 0

- Doesn't provide fault tolerance or redundancy.

When Raid 0 Should Be Used

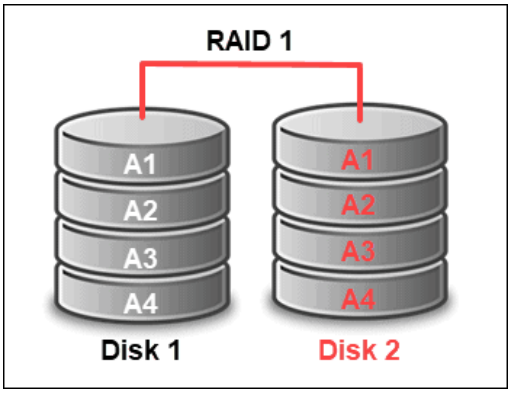
RAID 0 is used when performance is a priority and reliability is not. If you want to utilize your drives to the fullest and don't mind losing data, opt for RAID 0.

On the other hand, such a configuration does not necessarily have to be unreliable. You can set up disk striping on your system along with another RAID array that ensures data protection and redundancy.

RAID 1: Mirroring

RAID 1 is an array consisting of at least two disks where the same data is stored on each to ensure redundancy. The most common use of RAID 1 is setting up a mirrored pair consisting of two disks in which the contents of the first disk is mirrored in the second. This is why such a configuration is also called mirroring.

Unlike with RAID 0, where the focus is solely on speed and performance, the primary goal of RAID 1 is to provide redundancy. It eliminates the possibility of data loss and downtime by replacing a failed drive with its replica.



In such a setup, the array volume is as big as the smallest disk and operates as long as one drive is operational. Apart from reliability, mirroring enhances read performance as a request can be handled by any of the drives in the array. On the other hand, the write performance remains the same as with one disk and is equal to the slowest disk in the configuration.

Advantages of RAID 1

- Increased read performance.
- Provides redundancy and fault tolerance.
- Simple to configure and easy to use.

Disadvantages of RAID 1

- Uses only half of the storage capacity.
- More expensive (needs twice as many drivers).
- Requires powering down your computer to replace failed drive.

When Raid 1 Should Be Used

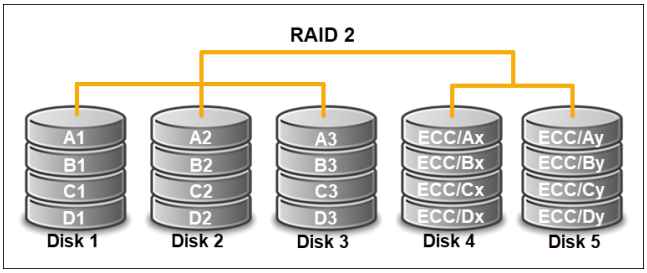
RAID 1 is used for mission-critical storage that requires a minimal risk of data loss. Accounting systems often opt for RAID 1 as they deal with critical data and require high reliability.

It is also suitable for smaller servers with only two disks, as well as if you are searching for a simple configuration you can easily set up (even at home).

Raid 2: Bit-Level Striping with Dedicated Hamming-Code Parity

RAID 2 is rarely used in practice today. It combines bit-level striping with error checking and information correction. This RAID implementation requires two groups of disks – one for writing the data and another for writing error correction codes. RAID 2 also requires a special controller for the synchronized spinning of all disks.

Instead of data blocks, RAID 2 stripes data at the bit level across multiple disks. Additionally, it uses the Humming error ode correction (ECC) and stores this information on the redundancy disk.



The array calculates the error code correction on the fly. While writing the data, it strips it to the data disk and writes the code to the redundancy disk. On the other hand, while reading data from the disk, it also reads from the redundancy disk to verify the data and make corrections if needed.

Advantages of RAID 2

- Reliability.
- The ability to correct stored information.

Disadvantages of RAID 2

- Expensive.
- Difficult to implement.
- Require entire disks for ECC.

When Raid 2 Should Be Used

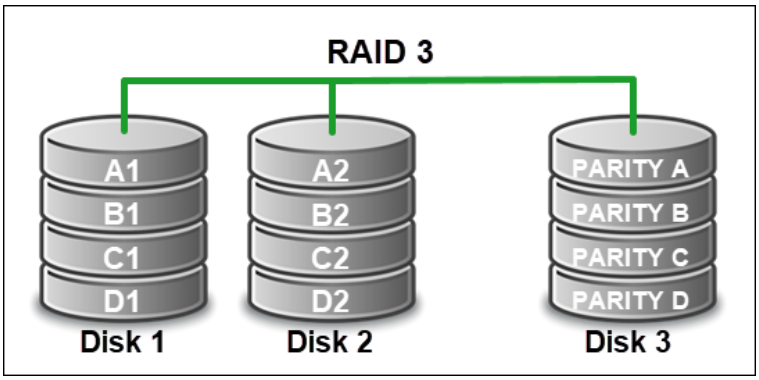
RAID 2 is not a common practice today as most of its features are now available on modern hard disks. Due to its cost and implementation requirements, this RAID level never became popular among developers.

Raid 3: Bit-Level Striping with Dedicated Parity

Like RAID 2, RAID 3 is rarely used in practice. This RAID implementation utilizes bit-level striping and a dedicated parity disk. Because of this, it requires at least three drives, where two are used for storing data strips, and one is used for parity.

To allow synchronized spinning, RAID 3 also needs a special controller. Due to its configuration and synchronized disk spinning, it achieves better

performance rates with sequential operations than random read/write operations.



Advantages of RAID 3

- Good throughput when transferring large amounts of data.
- High efficiency with sequential operations.
- Disk failure resiliency.

Disadvantages of RAID 3

- Not suitable for transferring small files.
- Complex to implement.
- Difficult to set up as software RAID.

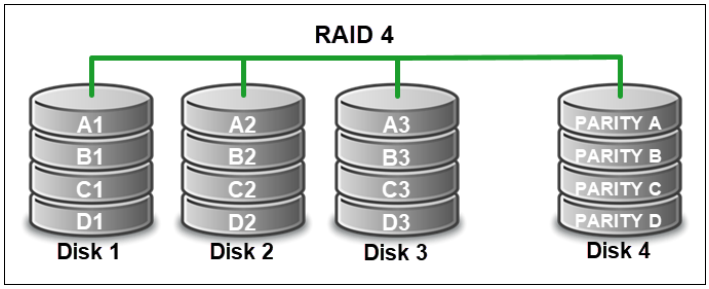
When Raid 3 Should Be Used

RAID 3 is not commonly used today. Its features are beneficial to a limited number of use cases requiring high transfer rates for long sequential reads and writes (such as video editing and production).

Raid 4: Block-Level Striping with Dedicated Parity

RAID 4 is another unpopular standard RAID level. It consists of block-level data striping across two or more independent diss and a dedicated parity disk.

The implementation requires at least three disks – two for storing data strips and one dedicated for storing parity and providing redundancy. As each disk is independent and there is no synchronized spinning, there is no need for a controller.



RAID 4 configuration is prone to bottlenecks when storing parity bits for each data block on a single drive. Such system bottlenecks have a large impact on system performance.

Advantages of RAID 4

- Fast read operations.
- Low storage overhead.
- Simultaneous I/O requests.

Disadvantages of RAID 4

- Bottlenecks that have big effect on overall performance.
- Slow write operations.
- Redundancy is lost if the parity disk fails.

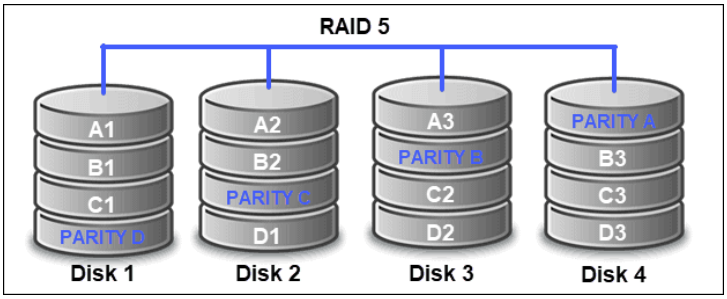
When Raid 4 Should Be Used

Considering its configuration, RAID 4 works best with use cases requiring sequential reading and writing data processes of huge files. Still, just like with RAID 3, in most solutions, RAID 4 has been replaced with RAID 5.

Raid 5: Striping with Parity

RAID 5 is considered the most secure and most common RAID implementation. It combines striping and parity to provide a fast and reliable setup. Such a configuration gives the user storage usability as with RAID 1 and the performance efficiency of RAID 0.

This RAID level consists of at least three hard drives (and at most, 16). Data is divided into data strips and distributed across different disks in the array. This allows for high performance rates due to fast read data transactions which can be done simultaneously by different drives in the array.



Parity bits are distributed evenly on all disks after each sequence of data has been saved. This feature ensures that you still have access to the data from parity bits in case of a failed drive. Therefore, RAID 5 provides redundancy through parity bits instead of mirroring.

Advantages of RAID 5

- High performance and capacity.
- Fast and reliable read speed.
- Tolerates single drive failure.

Disadvantages of RAID 5

- Longer rebuild time.
- Uses half of the storage capacity (due to parity).
- If more than one disk fails, data is lost.

- More complex to implement.

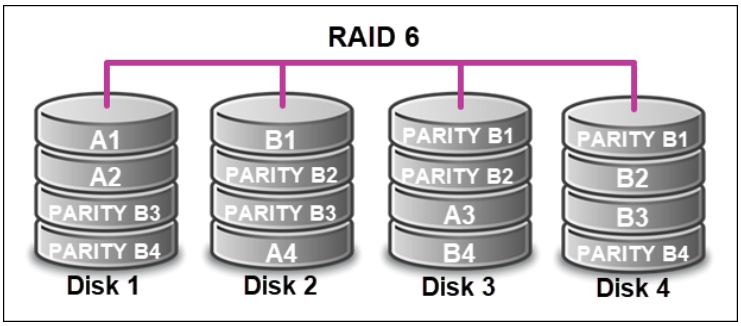
When Raid 5 Should Be Used

RAID 5 is often used for file and application servers because of its high efficiency and optimized storage. Additionally, it is the best, cost-effective solution if continuous data access is a priority and/or you require installing an operating system on the array.

Raid 6: Striping with Double Parity

RAID 6 is an array similar to RAID 5 with an addition of its double parity feature. For this reason, it is also referred to as the double-parity RAID.

This setup requires a minimum of four drives. The setup resembles RAID 5 but includes two additional parity blocks distributed across the disk. Therefore, it uses block-level striping to distribute the data across the array and stores two parity blocks for each data block.



Block-level striping with two parity blocks allows two disk failures before any data is lost. This means that in an event where two disks fail, RAID can still reconstruct the required data.

Its performance depends on how the array is implemented, as well as the total number of drives. Write operations are slower compared to other configurations due to its double parity feature.

Advantages of RAID 6

- High fault and drive-failure tolerance.
- Storage efficiency (when more than four drives are used).
- Fast read operations.

Disadvantages of RAID 6

- Rebuild time can take up to 24 hours.
- Slow write performance.
- Complex to implement.
- More expensive.

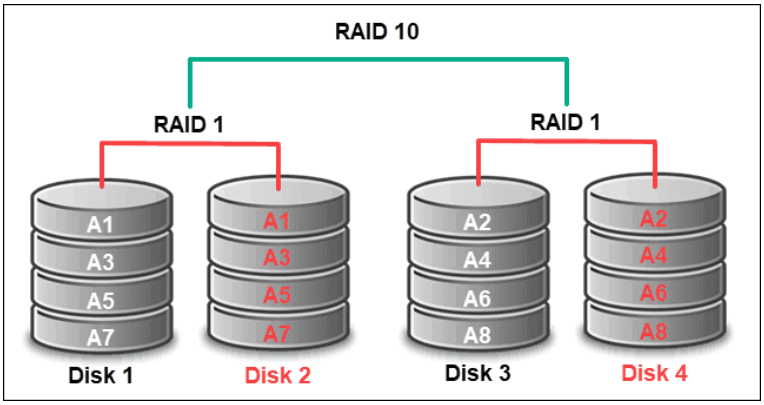
When Raid 6 Should Be Used

RAID 6 is a good solution for mission-critical applications where data loss cannot be tolerated. Therefore, it is often used for data management in defense sectors, healthcare, and banking.

Raid 10: Mirroring with Striping

RAID 10 is part of a group called nested or hybrid RAID, which means it is a combination of two different RAID levels. In the case of RAID 10, the array combines level 1 mirroring and level 0 striping. This RAID array is also known as RAID 1+0.

RAID 10 uses logical mirroring to write the same data on two or more drives to provide redundancy. If one disk fails, there is a mirrored image of the data stored on another disk. Additionally, the array uses block-level striping to distribute chunks of data across different drives. This improves performance and read and write speed as the data is simultaneously accessed from multiple disks.



To implement such a configuration, the array requires at least four drives, as well as a disk controller.

Advantages of RAID 10

- High performance.
- High fault-tolerance.
- Fast read and write operations.
- Fast rebuild time.

Disadvantages of RAID 10

- Limited scalability.
- Costly (compared to other RAID levels).
- Uses half of the disk space capacity.
- More complicated to set up.

When Raid 10 Should Be Used

RAID 10 is often used in use cases that require storing high volumes of data, fast read and write times, and high fault tolerance. Accordingly, this RAID level is often implemented for email servers, web hosting servers, and databases.

Non-Standard RAID

The RAID levels mentioned above are considered standard or commonly used RAID implementations. However, there is a myriad of ways you can set up redundant arrays of independent disks.

	Accordingly, many open-source projects and companies have created their own configurations to adhere to their needs. As a result, there are many non-standard RAID implementations, such as: • RAID-DP • Linux MD RAID 10 • RAID-Z • Drive Extender • Declustered RAID **Nested (Hybrid) RAID** You can combine two or more standard RAID levels to ensure better performance and redundancy. Such combinations are called nested (or hybrid) RAID levels. Hybrid RAID implementations are named after the RAID levels they incorporate. In most cases, they include two numbers where their order represents the layering scheme. Popular hybrid RAID levels include: • RAID 01 (striping and mirroring; also known as "mirror of stripes") • RAID 03 (byte-level striping and dedicated parity) • RAID 10 (disk mirroring and straight block-level striping) • RAID 50 (distributed parity and straight block-level striping) • RAID 60 (dual parity and straight block-level striping) • RAID 100 (a stripe of RAID 10s)			
5.	**Construct a B+ tree for the following set of search key values: Amala, Dolly, Bala, Sona, Chinna, Ezhil, Falley, Gilda. Assume the number of pointers in each node as 4. Also do the following operations Sequentially to the tree that you have constructed: Insert "Amutha", Insert "Raja", Insert "Vickram", Delete "Sona" and Delete "Gilda".**	K3	CO5	13
6.	**Experiment with the following schema structure and explain how it willbe stored in the physical memory. Employee (enonumber(5), ename varchar(15), manager_no number(5), dept char(5)) Also describe the method that is used to organize such type of records.**	K3	CO5	13
7.	**For the given schema structure Employee (enonumber(5), ename varchar(15), manager_no number(5), dept char(5)) by assuming 10 employees and 5 departments, construct the required primary index.**	K3	CO5	13
8.	**Suppose that we are using extendable hashing on a file that contains records with the following search key values 2,3,5,7,11,17,19,23,29,31. Show the extendable hash structure for this file, if the hash function is h(x)= x mod 8 and buckets can hold 3 records.**	K3	CO5	13
	PART C (Answer All Questions) (1 X 8 = 8)			
1.	**Consider the following ordering Schedule – 'S' of transactions:**	K3	CO4	8

	T1:R(A); T1:A:=A+5; T1; commit; T2:R(B); T2:B:=B+5; T3:R(C); T3:C:=C+5; T3:C:=C+5; T4:R(A); T4:A:=A+5; T4:R(D); T4:D:=D+5 ; T4:commit; T2:commit; T3:commit; **Let the initial value of A=B=C=D=0. The system follows log based recovery process of Immediate database modification. The assumptionis the concurrency control system uses strict 2PL, and all the transactions shares a common disk buffer and single log. Explain what happens during the recovery process, if there occurs a failure at 'T2: commit' statement.**			
2.	**Apply the log based recovery of your choice for the amount transfer from account 'A' to account 'B'.**	K3	CO4	8
3.	**Given below is the log information captured by the DBMS during the execution of the transactions.Analyses the log records and explain the recovery system carried by the DBMS. Also give the schedules that would create these log records.** **Log 1** <T_0 start> <T_0, A, 950> <T_0, B, 2050> <T_0 commit> <T_1 start> <T_1, C, 600> **Log 2** <T_0 start> <T_0, A, 1000, 950> <T_0, B, 2000, 2050> <T_0 commit> <T_1 start> <T_1, C, 700, 600> <T_1 commit>	K3	CO4	8
4.	**Utilize the concept of'Lost update', 'Uncommitted read', 'Non-repeatable read' and 'Phantom read' for demonstrating the amount transfer from account 'A' to account 'B'.**	K3	CO4	8

Course In-Charge **Module Coordinator** **HoD / CSE**