Q1: Describe RAID architecture. Explain RAID levels.
Answer:
RAID Architecture

RAID (Redundant Array of Independent Disks) is a storage technique that combines multiple physical hard disks into a single logical unit to achieve higher performance, data redundancy, and fault tolerance.

The RAID architecture consists of the following components:

RAID Controller
It manages the disk array and controls how data is written and read. It can be hardware-based or software-based.

Disk Array
A collection of two or more physical disks that operate together as one logical disk.

Data Management Techniques

Striping: Divides data into blocks and spreads them across disks to improve speed.

Mirroring: Copies the same data onto multiple disks to ensure data safety.

Parity: Stores error-checking information that helps recover data if a disk fails.

Logical View
The operating system views the RAID setup as a single disk, hiding internal complexity.

RAID architecture improves disk performance, reliability, or both, depending on the RAID level used.

RAID Levels
RAID 0 (Striping)

Data is split and stored across multiple disks.

Provides high performance.

No fault tolerance.

Disk failure causes total data loss.

RAID 1

(Mirroring)

Data is duplicated on two disks.

Provides high data reliability.

Storage capacity is reduced by half.

used where data safety is critical.

RAID 2

uses Hamming code for error correction.

Rarely used in practice due to complexity.

RAID 3

Data is striped at byte level with a dedicated parity disk.

Good for large sequential data transfers.

Parity disk can become a bottleneck.

RAID

4

Data is striped at block level with a dedicated parity disk.

Allows independent disk reads.

Parity disk bottleneck issue exists.

RAID 5

Data and parity are distributed across all disks.

Provides good performance and fault tolerance.

can tolerate failure of one disk.

widely used in servers.

RAID 6

Similar to RAID 5 but uses dual parity.

can tolerate failure of two disks.

More reliable but slightly slower write

performance.

RAID 10 (RAID 1 + RAID 0)

Combines mirroring and striping.

Offers high performance and high reliability.

Requires more disks and higher cost.

Q2: write a note on applications of indexing and hashing in databases.
Answer:
1. Indexing in Databases

Definition: Indexing is a technique used to improve the speed of data retrieval in a database table. It creates a separate data structure (index) that allows faster searching.

Applications:

Fast Query Processing – Speeds up SELECT queries, especially on large tables.

Efficient Sorting – Helps in ORDER By and GROUP By operations.

Enforcing Uniqueness – Indexes support constraints like PRIMARY KEY or UNIQUE.

Range Queries – Supports queries like WHERE age BETWEEN 20 AND 30 efficiently.

Join Optimization – Helps speed up JOIN operations between multiple tables.

2. Hashing in Databases

Definition: Hashing stores data using a hash function that maps a key to a specific location (bucket) in memory or disk.

Applications:

Fast Data Retrieval – Direct access using hash key is very fast.

Indexing Alternative – Hash-based indexing is useful for equality searches (WHERE id = 100).

Handling Large Datasets – useful in memory-resident tables or in distributed databases.

Load Balancing - Hashing distributes records evenly across storage or nodes.

Collision Resolution Techniques - Ensures data integrity even if multiple keys map to the same location.

Q3: Compare pessimistic and optimistic concurrency control
Answer:

Pessimistic Concurrency Control
Pessimistic concurrency control assumes that conflicts will happen so it locks the data before a transaction can access it. This prevents problems like lost updates or dirty reads, but it makes the system slower if many users are working at same time. Its mostly used in high contention situations like banking systems where multiple users may update same account. Because of locking, transactions may have to wait which can reduce performance, but data is safe.

Optimistic Concurrency Control
Optimistic concurrency control assumes conflicts are rare so it doesnt lock data while transaction is running. Instead it checks for conflicts only at commit time, and if there is a conflict the transaction is rolled back. This

makes it faster in low contention situations like online shopping carts where chances of 2 users updating same record is low. But if conflicts do happen, some work has to be redone which can be annoying.

Comparison
Overall, pessimistic is safer but slower because of locking, optimistic is faster but may need rollback if conflicts occur. Choice depends on the system requirements, like number of users and how often same data is accessed at same time.

Q4: Analyze the importance of recovery techniques in real-time database systems
Answer:

Importance of Recovery Techniques
Recovery techniques are very important in real-time database systems because these systems handle transactions that must meet strict timing constraints. If a system crashes or fails, data can get lost or become inconsistent, which can lead to wrong decisions or system failure. Recovery methods help restore the database to a correct state without violating timing requirements as much as possible.

Types of Failures Handled
Real-time databases can face several failures like system crash, media

failure, or transaction failure. Recovery techniques make sure that even if something goes wrong, completed transactions remain permanent (durability) and incomplete ones are rolled back, keeping the database consistent.

Impact on System Reliability
Without proper recovery methods, a real-time system cannot be trusted. For example, in air traffic control or medical monitoring systems, incorrect data or delayed recovery can have serious consequences. Techniques like checkpointing, log-based recovery, and shadow paging ensure the system can recover quickly and continue meeting real-time constraints.

Q5: Consider a relation R(A, B, C, D, E) with functional dependencies: A $\rightarrow$ B, BC $\rightarrow$ E, ED $\rightarrow$ A.

i) List all keys of R
ii) Is R in 3NF?
iii) Is R in BCNF?

Answer:

Step 1: Find candidate keys
Attributes: A, B, C, D,

E

FDs:

$A \rightarrow B$

$BC \rightarrow E$

$ED \rightarrow A$

we need a set of attributes that determines all attributes.

Start with ED: ED+ = {E, D}

$ED \rightarrow A \Rightarrow \{E, D, A\}$

$A \rightarrow B \Rightarrow \{E, D, A, B\}$

Need C to cover all attributes ⇒ EDC+ = {E, D, C, A, B}

So CDE is a candidate key.

Check smaller subsets:

ED alone → ED+ = {E, D, A, B}, missing C → Not key

Others like DC, EC, BC → Do not cover all attributes

Candidate key: CDE

Step 2: Check 3NF

3NF rule: For every FD $\varphi \rightarrow y$, one of these must hold:

$\varphi \rightarrow y$ is trivial

φ is a superkey

Every attribute in y is part of some candidate key

Check FDs:

A → B → A is not superkey, B not in key → violates 3NF

BC → E → BC not superkey, E is in key → OK

ED → A → ED not superkey, A not in key → violates 3NF

Conclusion: R is not in