Indexing and Hashing:

1. Basic Concepts:

Indexing:

- Indexing is a technique used to speed up the retrieval of records from a database or file system.
- An index is a data structure that holds a sorted list of key values and pointers to the corresponding records.

Hashing:

- Hashing is a technique to convert a key into a unique hash value using a hash function.
- The hash value is used to index into a hash table, where the actual data is stored.

2. Ordered Indices:

- Records in the index are arranged in a specific order.
- Enables efficient searching for exact matches and retrieval of data in a particular order.
- Examples: Binary search trees, B-trees.

3. Dense Index:

* A dense index has an index entry for every search key value in the data file.
* Each index entry points to the actual record in the data file.

4. Sparse Index:

* A sparse index has index entries only for some of the search key values.
* Typically, index entries point to the first record of a block of records.

5. primary and Secondary Index:

primary Index:

* Created on the primary key of a table.
* Ensures that each key value is unique.
* Often a clustered index, meaning the data is physically stored in the order of the primary key.

Secondary Index:

* Created on non-primary key columns.
* Can have duplicate key values.
* Typically a non-clustered index, meaning it holds pointers to the actual data rather than sorting.

6. B+-Tree Index Files:

* A B+-Tree is a balanced tree data structure that maintains sorted data and allows searches, sequential access, insertions and deletions in logarithmic time.

Structure:

* Internal nodes store keys and child pointers.
* Leaf nodes store keys and pointers to the actual data records.

7. Update on B+-Tree:

a. Insertion:

1. Find the appropriate leaf node where the key should be inserted.
2. Insert the key in stored order.
3. If the leaf node overflows.

b. Deletion:

1. Find the key in the leaf node.
2. Remove the key
3. If the leaf node underflows.

8. Static Hashing:

* In static hashing, the number of buckets is fixed.
* A has function maps keys to buckets.
* Collisions are handled using techniques like chaining or open addressing.

9. Dynamic Hashing:

* Dynamic hashing adjusts the number of buckets dynamically as the data grows.

10. Comparison of Ordered Indexing and Hashing:

Ordered Indexing	Hashing
suitable for range queries.	suitable for exact match queries.
maintains data in a stored order.	Does not maintain data in a stored order
can use structures like B+-Trees.	uses hash tables with hash functions.
slower for exact match queries compared to hashing.	Inefficient for range queries.

Concurrency Control and Deadlock:

1. What is a Transaction? Explain the ACID properties.

Transaction:

A transaction is a sequence of operations performed as a single logical unit of work. These operations could be a combination of data retrieval, updates, deletions, insertions.

ACID properties:

ACID properties guarantee that database transactions are processed reliably. The ACID properties are:

* Atomicity: Ensures that all operations within a transaction are completed; if not, the transaction is aborted.
* Consistency: Ensures that a transaction brings the database from one valid state to another valid state.
* Isolation: Ensures that transactions are executed in isolation from one another.
* Durability: Ensures that once a transaction has been committed, it will remain so, even in the event of a system failure.

2. Explain Various Locking Methods with examples.

⇒ Locking is used to control access to database resources in a concurrent environment. Different locking methods include:

- Shared & Exclusive Locks:

* Shared Lock (S): Allows multiple transaction to read a data item.

Example: If transaction T1 holds a shared lock on data item A, Transaction T2 can also obtain a shared lock on A, allowing both to read A concurrently.

* Exclusive Lock (X): Allows only one transaction to write to a data item.

Example: If Transaction T1 holds an exclusive lock on data item A, no other transaction can obtain any type of lock on A until T1 releases the exclusive lock.

3. Define Concurrency Control. Explain Different Concurrency Control Methods.

⇒ Concurrency Control: Concurrency control refers to the mechanisms used to manage concurrent access to a database by multiple transactions, guaranteeing data integrity and consistency.

* Different Concurrency Control Methods:

* Locking: As discussed above, locking ensures exclusive or shared access to data, preventing inconsistencies.

* Optimistic Concurrency Control (OCC): Transactions proceed without acquiring locks initially. At commit time, the system validates if any uncommitted changes create conflicts.

* Timestamp Ordering: Transactions are assigned timestamps upon initiation. Conflicts are resolved based on timestamps, ensuring a defined order of execution.

5. Deadlocks:

A deadlock occurs when two or more transactions are waiting for each other to release resources, causing all of them to be unable to proceed.

a. Starvation: A specific transaction is continuously rolled back due to deadlocks involving higher-priority transactions.

b. Deadlock Handling:

Methods to handle deadlocks include deadlock prevention, detection and recovery.

c. Deadlock prevention: Techniques to avoid deadlocks altogether:

* Ordering: Imposing a fixed order on resource acquisition
* Timestamp Ordering: Assigning timestamps to transactions and handling conflicts based on these timestamps.

d. Deadlock Detection: Regularly check for cycles in the wait-for graph, where nodes represent transaction and edges represent waiting for resources.

e. Deadlock Recovery: Once a deadlock is detected recovery involves rolling back a transaction and restarting it.

Database System Architectures

1. Centralized and Client-Server Systems:

Centralized Systems:

* All database components (DBMS, data storage and user interfaces) reside on a single machine.
* Common in early database systems.

Client-Server Systems:

* Split into two main components: Clients and servers.
* Client: Handles the user interface and sends requests to the server.
* Server: Manages data processing, storage and query execution.

2. Server System Architectures:

Transaction Server:

* Also known as OLTP (online Transaction processing) server.
* Specializes in managing and ensuring the ACID properties of transactions.

Data Server:

* Manages the storage and retrieval of data.
* Can handle both OLTP and OLAP (online Analytical processing) workloads.

3. parallel Systems

- **parallel systems:**

* Use multiple processors or machines to perform database operations in parallel, enhancing performance and capacity.

- **Speedup:**

* The Speedup is the improvement in performance that is achieved by using multiple processors. It is measured by the execution time of a query on a single processor to the execution time of the same query on multiple processors.

- **Scaleup:**

* Scaleup is the improvement in performance that is achieved by increasing the resources of a single processor, such as increasing the clock speed or adding more memory.

4. Distributed Systems

- **Distributed Systems:**

* A distributed database system is a database system that is spread across multiple computers.

Trade-offs in Distributed Systems:

* Latency vs. Consistency: Ensuring data consistency across locations can increase latency.
* Complexity vs. Flexibility: More complex to manage but offers greater flexibility and fault tolerance.
* Resource sharing vs. Autonomy: Allows resource sharing but can complicate site autonomy.

Implementation Issues for Distributed Databases:

* Data Distribution: Deciding how to distribute data across sites (e.g. fragmentation, replication)
* Replication: Managing consistency and updates across replicated data.
* Concurrency Control: Coordinating concurrent transactions across sites.
* Fault Tolerance: Ensuring system reliability and recovery from site or network failures.
* Query processing: Optimizing queries that access data from multiple sites.