MAHARNA PRATAP GROUP OF INSTITUTIONS, KANPUR

DBMS Interview Questions and Answers
120 Questions

1. Basics

1. **Q: What is a DBMS?**
   - **A:** A Database Management System (DBMS) is a software system designed to manage and organize data in a database. It provides functionalities for storing, retrieving, updating, and managing data, ensuring data integrity, security, and consistency.
2. **Q: What are the advantages of using a DBMS?**
   - **A:** Advantages include data redundancy control, data consistency, data sharing, data integrity, data security, improved data access, better decision-making, and backup and recovery facilities.
3. **Q: What is a database?**
   - **A:** A database is an organized collection of structured information, or data, typically stored electronically in a computer system. It is designed to efficiently store and retrieve large amounts of data.
4. **Q: Differentiate between data and information.**
   - **A:** Data are raw, unorganized facts and figures (e.g., a list of numbers). Information is processed, organized, and structured data that provides context and meaning (e.g., the average of those numbers).
5. **Q: What is a schema in DBMS?**
   - **A:** A schema is the logical design or structure of a database. It defines how data is organized and how the relations among them are associated. It's a blueprint of the database.
6. **Q: Explain the three-schema architecture.**
   - **A:** It divides the database into three levels:
     - **External Schema (View Level):** How users see the data.
     - **Conceptual Schema (Logical Level):** Describes what data is stored and how it's related, independent of physical storage.
     - **Internal Schema (Physical Level):** Describes how data is physically stored on disk.
7. **Q: What is data independence?**
   - **A:** Data independence is the ability to modify the schema at one level of the database system without affecting the schema at the next higher level.
     - **Physical Data Independence:** Changes in the internal schema do not affect the conceptual schema.

        - **Logical Data Independence:** Changes in the conceptual schema do not affect the external schema.
8. **Q: What is a Data Model? List some common data models.**
    - **A:** A data model is a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints. Common models include Relational Model, Hierarchical Model, Network Model, and Object-Oriented Model.
9. **Q: What is the Relational Model?**
    - **A:** The relational model organizes data into one or more tables (or relations) of rows and columns. Each row represents a record, and each column represents an attribute. It's the most widely used data model.
10. **Q: What is a Tuple and an Attribute?**
    - **A:** In the relational model, a **Tuple** refers to a row in a table, representing a single record. An **Attribute** refers to a column in a table, representing a specific characteristic or property of the entity.
11. **Q: What is a Domain in DBMS?**
    - **A:** A domain is the set of all possible values for an attribute. For example, the domain for an 'Age' attribute might be integers between 0 and 120.
12. **Q: What is a Data Dictionary?**
    - **A:** A data dictionary is a centralized repository of information about data, such as meaning, relationships to other data, origin, usage, and format. It's metadata (data about data).
13. **Q: What is a Transaction?**
    - **A:** A transaction is a single logical unit of work that accesses and possibly modifies the contents of a database. It must be executed completely or not at all (atomicity).
14. **Q: What is DDL, DML, DCL, and TCL?**
    - **A:**
        - **DDL (Data Definition Language):** Used to define and modify the database structure (e.g., CREATE, ALTER, DROP).
        - **DML (Data Manipulation Language):** Used to manage data within schema objects (e.g., SELECT, INSERT, UPDATE, DELETE).
        - **DCL (Data Control Language):** Used to control access to data (e.g., GRANT, REVOKE).
        - **TCL (Transaction Control Language):** Used to manage transactions (e.g., COMMIT, ROLLBACK, SAVEPOINT).
15. **Q: What is an Entity-Relationship (ER) Model?**
    - **A:** The ER model is a high-level conceptual data model that describes the data requirements and relationships in a system using entities, attributes, and relationships. It's often used for database design.
16. **Q: What is an Entity, Entity Set, and Attribute in ER Model?**
    - **A:** An **Entity** is a real-world object or concept (e.g., Student, Course). An **Entity Set** is a collection of similar entities (e.g., all students). An

Attribute is a property that describes an entity (e.g., Student ID, Student Name).

17. **Q: What are the different types of relationships in an ER Model?**
    - **A:**
        - **One-to-One (1:1):** Each entity in one set is related to at most one entity in another set.
        - **One-to-Many (1:M):** One entity in the first set can be related to multiple entities in the second set, but an entity in the second set can only be related to one in the first.
        - **Many-to-One (M:1):** Multiple entities in the first set can be related to one entity in the second set, but an entity in the second set can be related to multiple in the first.
        - **Many-to-Many (M:N):** Multiple entities in the first set can be related to multiple entities in the second set.

2. Keys

1. **Q: What is a Key in DBMS?**
    - **A:** A key is an attribute or a set of attributes that uniquely identifies a row (tuple) in a table (relation). Keys are crucial for establishing relationships between tables.
2. **Q: What is a Super Key?**
    - **A:** A Super Key is an attribute or a set of attributes that can uniquely identify a tuple in a relation. It can contain extra attributes that are not necessary for unique identification.
3. **Q: What is a Candidate Key?**
    - **A:** A Candidate Key is a minimal Super Key. It's a Super Key from which no attribute can be removed without losing the uniqueness property. A table can have multiple candidate keys.
4. **Q: What is a Primary Key?**
    - **A:** A Primary Key is a Candidate Key chosen by the database designer to uniquely identify each tuple in a relation. It must contain unique values and cannot have NULL values.
5. **Q: What is an Alternate Key?**
    - **A:** An Alternate Key is a Candidate Key that is not chosen as the Primary Key. If there are multiple candidate keys, one is chosen as primary, and the others become alternate keys.
6. **Q: What is a Foreign Key?**
    - **A:** A Foreign Key is an attribute or a set of attributes in one table that refers to the Primary Key of another table. It establishes a link between two tables, enforcing referential integrity.
7. **Q: Explain the purpose of a Foreign Key.**
    - **A:** The purpose of a Foreign Key is to enforce referential integrity, meaning that if a foreign key value exists in the referencing table, it must

have a corresponding primary key value in the referenced table. It links related data across tables.

8. **Q: What is a Composite Key?**
    - **A:** A Composite Key is a key that consists of two or more attributes that together uniquely identify a row in a table. No single attribute in the composite key can uniquely identify the row on its own.
9. **Q: What is a Unique Key? How is it different from a Primary Key?**
    - **A:** A Unique Key constraint ensures that all values in a column (or a set of columns) are unique.
    - **Difference:** A Primary Key implicitly creates a unique index and does not allow NULL values. A Unique Key allows one NULL value. A table can have only one Primary Key but multiple Unique Keys.
10. **Q: What is a Surrogate Key?**
    - **A:** A Surrogate Key is an artificial key (typically an auto-incrementing integer) that is generated by the database system to uniquely identify each record. It has no intrinsic meaning to the business data.
11. **Q: When would you use a Surrogate Key over a Natural Key?**
    - **A:** Use a Surrogate Key when a natural key is too long, complex, mutable, or non-existent. It simplifies joins, improves performance, and provides stability if natural key attributes change.
12. **Q: What is an Index in DBMS?**
    - **A:** An index is a data structure that improves the speed of data retrieval operations on a database table. It works like an index in a book, allowing the database system to quickly locate rows without scanning the entire table.
13. **Q: Differentiate between Clustered and Non-Clustered Index.**
    - **A:**
        - **Clustered Index:** Determines the physical order of data storage in a table. A table can have only one clustered index. Data rows are stored in the order of the clustered index key.
        - **Non-Clustered Index:** Does not alter the physical order of table rows. It creates a separate structure containing the indexed columns and pointers to the actual data rows. A table can have multiple non-clustered indexes.
14. **Q: What is a Hash Key?**
    - **A:** A hash key is a value generated by a hash function from a data item. It's used in hashing to quickly locate data records in a hash table.
15. **Q: What is an Inverted Index?**
    - **A:** An inverted index is a database index that stores a mapping from content (like words or numbers) to its locations in a database table. It's commonly used in full-text search engines.
16. **Q: Can a table have multiple foreign keys?**
    - **A:** Yes, a table can have multiple foreign keys, each referencing the primary key of a different table, or even the same table multiple times

(e.g., an employee table having foreign keys for manager_id and supervisor_id, both referencing employee_id).

17. **Q: What is a Candidate Key used for?**
    - **A:** Candidate keys are used to identify all possible attributes or sets of attributes that can uniquely identify a tuple. From these, one is chosen as the primary key, and the others become alternate keys.

3. Dependencies

1. **Q: What is a Functional Dependency (FD)?**
    - **A:** A functional dependency (X →Y) means that the value of attribute set X uniquely determines the value of attribute set Y. If two tuples have the same X value, they must also have the same Y value.
2. **Q: What is a Trivial Functional Dependency?**
    - **A: A functional dependency X → Y is trivial if Y is a subset of X (Y ⊆ X). For example, {StudentID, Name} → Name is trivial because Name is part of {StudentID, Name}.**
3. **Q: What is a Non-Trivial Functional Dependency?**
    - **A:** A functional dependency X → Y is non-trivial if Y is not a subset of X (Y ⊄ X). For example, StudentID → Name is non-trivial.
4. **Q: What is a Full Functional Dependency?**
    - **A:** A functional dependency X → Y is fully functionally dependent if Y depends on all attributes in X, and not on any proper subset of X. This is important for 2NF.
5. **Q: What is a Partial Dependency?**
    - **A:** A partial dependency occurs when a non-prime attribute (an attribute not part of any candidate key) is functionally dependent on only a part of a candidate key, not the entire key. This violates 2NF.
6. **Q: What is a Transitive Dependency?**
    - **A:** A transitive dependency occurs when a non-prime attribute is functionally dependent on another non-prime attribute, which in turn is functionally dependent on the primary key. If A → B and B → C, then A → C is a transitive dependency. This violates 3NF.
7. **Q: What is Multivalued Dependency (MVD)?**
    - **A:** A multivalued dependency (X →→ Y) means that for a given value of X, there is a set of zero or more associated Y values, and this set of Y values is independent of any other attributes in the relation. This violates 4NF.
8. **Q: What is Join Dependency?**
    - **A:** A join dependency exists if a relation R can be reconstructed by joining several of its projections. It is a generalization of multivalued dependency. This relates to 5NF.
9. **Q: What are Armstrong's Axioms? List them.**
    - **A:** Armstrong's Axioms are a set of inference rules used to infer all functional dependencies on a relational database. They are:

        - **Reflexivity:** If $Y \subseteq X$, then $X \rightarrow Y$.
        - **Augmentation:** If $X \rightarrow Y$, then $XZ \rightarrow YZ$ (where Z is any set of attributes).
        - **Transitivity:** If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$.
10. **Q: What is the Closure of a set of Functional Dependencies (F+)?**
    - **A:** The closure of a set of functional dependencies F (denoted F+) is the set of all functional dependencies that can be logically inferred from F using Armstrong's Axioms.
11. **Q: What is the Closure of an Attribute Set (X+)?**
    - **A:** The closure of an attribute set X (denoted X+) with respect to a set of functional dependencies F is the set of all attributes that are functionally dependent on X. It helps find candidate keys.
12. **Q: How do you find a Candidate Key using functional dependencies?**
    - **A:** To find a candidate key, you need to find an attribute set X such that its closure (X+) contains all attributes of the relation, and no proper subset of X has this property.
13. **Q: What is a Prime Attribute and Non-Prime Attribute?**
    - **A:** A **Prime Attribute** is an attribute that is part of *any* candidate key. A **Non-Prime Attribute** is an attribute that is not part of *any* candidate key.
14. **Q: What is the significance of functional dependencies in database design?**
    - **A:** Functional dependencies are fundamental for database normalization. They help identify data redundancies and anomalies, guiding the decomposition of relations into a well-structured, normalized form.
15. **Q: How does a partial dependency lead to anomalies?**
    - **A:** If a non-prime attribute depends on only part of a composite primary key, it can lead to:
        - **Insertion Anomaly:** Cannot insert part of a record without the full key.
        - **Deletion Anomaly:** Deleting a record might unintentionally delete information about other entities.
        - **Update Anomaly:** Updating a value requires updating multiple rows, leading to inconsistency if one is missed.
16. **Q: How does a transitive dependency lead to anomalies?**
    - **A:** If a non-prime attribute depends on another non-prime attribute, which depends on the primary key, it leads to:
        - **Insertion Anomaly:** Cannot record information about the intermediate non-prime attribute without the primary key.
        - **Deletion Anomaly:** Deleting a record might lose information about the intermediate non-prime attribute.
        - **Update Anomaly:** Updating the intermediate non-prime attribute's value requires updating multiple rows.
17. **Q: What is a Lossless-Join Decomposition?**

- **A:** A decomposition of a relation R into R1, R2, ..., Rn is lossless-join if the natural join of R1, R2, ..., Rn yields exactly the original relation R. This means no spurious tuples are generated and no information is lost.
-

4. Normalization

1. **Q: What is Normalization in DBMS?**
   - **A:** Normalization is a systematic process of organizing the columns and tables of a relational database to minimize data redundancy and improve data integrity. It involves decomposing tables into smaller, well-structured tables.
2. **Q: What are the main goals of Normalization?**
   - **A:**
     - Eliminate redundant data.
     - Ensure data dependencies make sense (data is stored logically).
     - Reduce data anomalies (insertion, update, deletion).
     - Improve data integrity and consistency.
3. **Q: What is 1NF (First Normal Form)?**
   - **A:** A relation is in 1NF if all attributes contain atomic (indivisible) values, and there are no repeating groups of columns. Each column should contain a single value for each row.
4. **Q: What is 2NF (Second Normal Form)?**
   - **A:** A relation is in 2NF if it is in 1NF and all non-prime attributes are fully functionally dependent on the entire primary key. It eliminates partial dependencies.
5. **Q: What is 3NF (Third Normal Form)?**
   - **A:** A relation is in 3NF if it is in 2NF and there are no transitive dependencies. That is, no non-prime attribute is dependent on another non-prime attribute.
6. **Q: What is BCNF (Boyce-Codd Normal Form)?**
   - **A:** A relation is in BCNF if for every non-trivial functional dependency X -> Y, X is a superkey. BCNF is a stricter form of 3NF. It handles cases where 3NF might still have anomalies (e.g., when a non-prime attribute determines a part of a candidate key).
7. **Q: When is BCNF preferred over 3NF?**
   - **A:** BCNF is preferred over 3NF when there are multiple overlapping candidate keys, or when a non-prime attribute determines a part of a candidate key. BCNF ensures that all determinants are candidate keys.
8. **Q: What is 4NF (Fourth Normal Form)?**
   - **A:** A relation is in 4NF if it is in BCNF and contains no multi-valued dependencies. It addresses situations where multiple independent multi-valued facts about an entity are stored in the same table.
9. **Q: What is 5NF (Fifth Normal Form) / Project-Join Normal Form (PJNF)?**

- **A:** A relation is in 5NF if it is in 4NF and contains no join dependencies. It aims to eliminate redundancy that can arise from complex join dependencies. It's rarely used in practice.

10. **Q: What is Denormalization? When is it used?**
    - **A:** Denormalization is the process of intentionally introducing redundancy into a database by combining tables or adding duplicate data. It is used to improve query performance, especially in data warehousing or reporting systems, at the cost of increased redundancy and potential for anomalies.
11. **Q: What are the trade-offs of Normalization?**
    - **A:**
      - **Pros:** Reduces data redundancy, improves data integrity, saves storage space, easier to maintain.
      - **Cons:** Increased number of tables, more complex queries (due to joins), potentially slower read performance (due to joins).
12. **Q: Explain the process of normalizing a table from 1NF to 2NF.**
    - **A:**

1. Ensure the table is in 1NF.
2. Identify the primary key.
3. For any non-prime attribute that is partially dependent on the primary key, remove it and the part of the primary key it depends on to a new table. The partial key becomes the primary key of the new table, and a foreign key in the original table.

13. **Q: Explain the process of normalizing a table from 2NF to 3NF.**
    - **A:**

0. Ensure the table is in 2NF.
1. Identify any transitive dependencies (where a non-prime attribute depends on another non-prime attribute).
2. Remove the transitively dependent attributes and the attributes they depend on to a new table. The determinant non-prime attribute becomes the primary key of the new table, and a foreign key in the original table.

14. **Q: What are update anomalies, and how does normalization help prevent them?**
    - **A:** Update anomalies occur when the same information is stored in multiple places, and an update to one instance is not propagated to all instances, leading to inconsistencies. Normalization reduces redundancy, ensuring that each fact is stored in only one place, thus preventing update anomalies.
15. **Q: What are insertion anomalies, and how does normalization help prevent them?**
    - **A:** Insertion anomalies occur when certain data cannot be inserted into the database without the presence of other data. Normalization decomposes tables, allowing independent facts to be inserted without requiring unrelated information.
16. **Q: What are deletion anomalies, and how does normalization help prevent them?**

- **A:** Deletion anomalies occur when deleting a record unintentionally deletes other related information. Normalization separates data into distinct tables based on their dependencies, so deleting one type of information does not affect unrelated information.

17. **Q: Is it always necessary to normalize a database to 3NF or BCNF?**
    - **A:** Not always. While higher normal forms reduce redundancy and anomalies, they can lead to more complex queries and slower performance due to increased joins. The choice of normalization level depends on the specific application's requirements, performance considerations, and data characteristics. Sometimes, denormalization is intentionally applied for performance.

5. Transaction Processing

1. **Q: What is a Transaction in DBMS?**
    - **A:** A transaction is a single logical unit of work that accesses and possibly modifies the contents of a database. It is a sequence of operations (read, write) that are treated as a single, indivisible unit.
2. **Q: What are the ACID properties of a transaction? Explain each.**
    - **A:** ACID is an acronym for properties guaranteeing that database transactions are processed reliably.
        - **Atomicity:** All operations within a transaction are completed successfully, or none are. It's an "all or nothing" principle.
        - **Consistency:** A transaction brings the database from one valid state to another valid state, preserving all defined rules and constraints.
        - **Isolation:** Concurrent transactions execute in isolation from each other. The intermediate state of one transaction is not visible to other concurrent transactions.
        - **Durability:** Once a transaction is committed, its changes are permanent and survive system failures (e.g., power outages, crashes).
3. **Q: What is a Commit operation?**
    - **A:** A Commit operation makes all changes performed by a transaction permanent in the database. Once committed, the changes cannot be undone by the transaction itself.
4. **Q: What is a Rollback operation?**
    - **A:** A Rollback operation undoes all changes made by a transaction since its beginning. It restores the database to the state it was in before the transaction started. This is used if a transaction fails or is aborted.
5. **Q: What is a Savepoint?**
    - **A:** A Savepoint is a point within a transaction that can be rolled back to, without rolling back the entire transaction. It allows for partial rollbacks within a larger transaction.
6. **Q: What is the purpose of a transaction log (journal)?**

- **A:** A transaction log (or journal) is a sequential record of all changes made to the database by transactions. It's crucial for recovery operations, allowing the database to be restored to a consistent state after a crash (using redo/undo operations).

7. **Q: What is the difference between a logical log and a physical log?**
   - **A:**
     - **Logical Log:** Records the operations performed (e.g., "update row X, set column Y to Z").
     - **Physical Log:** Records the actual before and after images of the data blocks that were changed.
8. **Q: What is a schedule in transaction processing?**
   - **A:** A schedule (or history) is a sequence of operations by a set of concurrent transactions. It defines the order in which the operations of the concurrent transactions are executed.
9. **Q: What is a serial schedule?**
   - **A:** A serial schedule is one where the operations of each transaction are executed consecutively, without any interleaving of operations from other transactions. All transactions run one after another.
10. **Q: What is a serializable schedule?**
    - **A:** A serializable schedule is a non-serial schedule that produces the same final state of the database as some serial schedule. It ensures that concurrent execution is equivalent to some serial execution, thus preserving consistency.
11. **Q: What are the different types of serializability?**
    - **A:**
      - **Conflict Serializability:** A schedule is conflict serializable if it can be transformed into a serial schedule by swapping non-conflicting operations.
      - **View Serializability:** A schedule is view serializable if its final state is the same as some serial schedule, considering read and write operations. (More complex to check than conflict serializability).
12. **Q: What is a cascading rollback (cascading abort)?**
    - **A:** A cascading rollback occurs when the failure of one transaction causes several other dependent transactions to also abort and rollback. This happens if transactions read data written by an uncommitted transaction that later aborts.
13. **Q: How can cascading rollbacks be avoided?**
    - **A:** Cascading rollbacks can be avoided by using strict two-phase locking (2PL) or by ensuring that a transaction commits only after all transactions that have read its uncommitted data have also committed.
14. **Q: What is a recoverable schedule?**
    - **A:** A recoverable schedule is one where, if a transaction T commits, then all transactions that have read data written by T must also commit. This

prevents a committed transaction from having read data from an aborted transaction.

15. **Q: What is a strict schedule?**
    - **A:** A strict schedule is a recoverable schedule where a transaction can neither read nor write an item X until the transaction that last wrote X has committed or aborted. This prevents both cascading rollbacks and uncommitted dependencies.
16. **Q: What is a Phantom Read?**
    - **A:** A phantom read occurs when, during a transaction, a query is executed twice, and the second execution returns a different set of rows. This happens because another concurrent transaction inserted new rows that meet the query's criteria between the two executions.
17. **Q: What is a Dirty Read (Uncommitted Read)?**
    - **A:** A dirty read occurs when a transaction reads data that has been written by another transaction that has not yet been committed. If the uncommitted transaction later rolls back, the first transaction will have read "dirty" or invalid data.

6. Concurrency Control

1. **Q: What is Concurrency Control in DBMS?**
    - **A:** Concurrency control is the process of managing simultaneous operations on a database without having them interfere with one another. Its goal is to ensure data integrity and consistency while maximizing throughput and response time.
2. **Q: Why is Concurrency Control necessary?**
    - **A:** It's necessary to prevent problems that arise from concurrent access, such as:
        - Lost Update Problem
        - Dirty Read Problem
        - Unrepeatable Read Problem
        - Phantom Read Problem
        - Incorrect Summary Problem
3. **Q: Explain the Lost Update Problem.**
    - **A:** The lost update problem occurs when two concurrent transactions read the same data item, then both update it, but the update of one transaction overwrites (and thus "loses") the update of the other.
4. **Q: Explain the Unrepeatable Read Problem.**
    - **A:** An unrepeatable read occurs when a transaction reads the same data item twice, but the data item's value changes between the two reads because another committed transaction updated it.
5. **Q: What is a Lock?**
    - **A:** A lock is a mechanism used in concurrency control to restrict access to a data item by concurrent transactions. When a transaction acquires a

lock on a data item, other transactions are prevented from accessing it in a conflicting way.

6. **Q: Differentiate between Shared Lock (S-Lock) and Exclusive Lock (X-Lock).**
    - **A:**
        - **Shared Lock (S-Lock/Read Lock):** Allows multiple transactions to read the same data item concurrently. No transaction can acquire an X-lock on the item while S-locks exist.
        - **Exclusive Lock (X-Lock/Write Lock):** Allows only one transaction to read and write a data item at a time. No other transaction can acquire any type of lock (S or X) on the item.
7. **Q: What is Two-Phase Locking (2PL)?**
    - **A:** Two-Phase Locking (2PL) is a concurrency control protocol that ensures serializability. It has two phases:
        - **Growing Phase:** A transaction can acquire new locks but cannot release any.
        - **Shrinking Phase:** A transaction can release existing locks but cannot acquire any new locks.
8. **Q: What is Strict Two-Phase Locking (Strict 2PL)?**
    - **A:** Strict 2PL is a stricter version of 2PL where all exclusive (write) locks are held until the transaction commits or aborts. This prevents dirty reads and cascading rollbacks.
9. **Q: What is Deadlock? How can it be prevented or detected?**
    - **A:** Deadlock occurs when two or more transactions are waiting indefinitely for each other to release locks.
        - **Prevention:** Pre-claiming locks, ordering resources, timestamp ordering.
        - **Detection:** Using a wait-for graph (a cycle in the graph indicates a deadlock), then choosing a victim to abort.
10. **Q: Explain Timestamp-Based Concurrency Control.**
    - **A:** In timestamp-based concurrency control, each transaction is assigned a unique timestamp. The system ensures that the effective serial execution order of transactions is the same as their timestamp order. It uses read-timestamps (RTS) and write-timestamps (WTS) for data items.
11. **Q: What is Optimistic Concurrency Control (OCC)?**
    - **A:** Optimistic concurrency control assumes that conflicts are rare. Transactions execute without acquiring locks during their read phase. They validate their changes at the commit phase. If validation fails, the transaction is rolled back and restarted.
12. **Q: Differentiate between Pessimistic and Optimistic Concurrency Control.**
    - **A:**
        - **Pessimistic:** Assumes conflicts are frequent, uses locking to prevent them *before* they occur. Good for high contention.

        - **Optimistic:** Assumes conflicts are rare, allows transactions to proceed and checks for conflicts *at commit time*. Good for low contention.

13. **Q: What are Isolation Levels in SQL? List them.**
    - **A:** Isolation levels define how and when changes made by one operation become visible to other concurrent operations. They balance consistency with concurrency.
        - **READ UNCOMMITTED:** Allows dirty reads, non-repeatable reads, phantom reads.
        - **READ COMMITTED:** Prevents dirty reads, allows non-repeatable reads, phantom reads.
        - **REPEATABLE READ:** Prevents dirty reads, non-repeatable reads, allows phantom reads.
        - **SERIALIZABLE:** Prevents dirty reads, non-repeatable reads, and phantom reads. Highest isolation, lowest concurrency.
14. **Q: Which isolation level is the default in most DBMS?**
    - **A:** READ COMMITTED is the default isolation level in most commercial DBMS (e.g., SQL Server, Oracle, PostgreSQL).
15. **Q: What is a Deadlock Detection Algorithm?**
    - **A:** A common algorithm involves constructing a "wait-for graph" where nodes are transactions and an edge T1 -> T2 exists if T1 is waiting for T2 to release a lock. A cycle in this graph indicates a deadlock.
16. **Q: What is a Livelock?**
    - **A:** Livelock is a situation where two or more transactions repeatedly change their state in response to each other without making any actual progress. Unlike deadlock, transactions are not blocked but are actively changing their state, yet cannot complete their work.
17. **Q: How does a DBMS recover from a system crash?**
    - **A:** DBMS uses logging and checkpointing. After a crash, it performs:
        - **Redo operations:** Apply changes from committed transactions that were not yet written to disk.
        - **Undo operations:** Rollback changes from uncommitted transactions that were written to disk. This restores the database to a consistent state.

7. SQL/Joins

1. **Q: What is SQL?**
    - **A:** SQL (Structured Query Language) is a standard language for managing and manipulating relational databases. It's used for defining, querying, and controlling data.
2. **Q: What are the main types of SQL commands?**
    - **A:** DDL (Data Definition Language), DML (Data Manipulation Language), DCL (Data Control Language), TCL (Transaction Control Language).

3. **Q: What is the difference between DELETE, TRUNCATE, and DROP?**
    - **A:**
        - **DELETE (DML):** Removes rows from a table based on a WHERE clause. It's a logged operation, can be rolled back, and fires triggers.
        - **TRUNCATE (DDL):** Removes all rows from a table, but keeps the table structure. It's faster than DELETE (no row-by-row logging), cannot be rolled back, and does not fire triggers. Resets identity columns.
        - **DROP (DDL):** Removes the entire table (structure and data) from the database. Cannot be rolled back.
4. **Q: What is a JOIN in SQL?**
    - **A:** A JOIN clause is used to combine rows from two or more tables, based on a related column between them.
5. **Q: Explain INNER JOIN.**
    - **A:** INNER JOIN returns only the rows where there is a match in *both* tables based on the join condition. It's the most common type of join.
6. **Q: Explain LEFT JOIN (or LEFT OUTER JOIN).**
    - **A:** LEFT JOIN returns all rows from the left table, and the matching rows from the right table. If there's no match in the right table, NULL values are returned for columns from the right table.
7. **Q: Explain RIGHT JOIN (or RIGHT OUTER JOIN).**
    - **A:** RIGHT JOIN returns all rows from the right table, and the matching rows from the left table. If there's no match in the left table, NULL values are returned for columns from the left table.
8. **Q: Explain FULL JOIN (or FULL OUTER JOIN).**
    - **A:** FULL JOIN returns all rows when there is a match in either the left or the right table. If there's no match, NULL values are returned for columns from the non-matching side.
9. **Q: What is a CROSS JOIN?**
    - **A:** A CROSS JOIN produces a Cartesian product of the two tables involved. It returns every possible combination of rows from the first table with every row from the second table.
10. **Q: What is a SELF JOIN?**
    - **A:** A SELF JOIN is a join in which a table is joined with itself. This is useful when you need to compare rows within the same table, typically by using aliases for the table.
11. **Q: What is the difference between WHERE and HAVING clauses?**
    - **A:**
        - **WHERE:** Filters individual rows *before* grouping. It cannot use aggregate functions.
        - **HAVING:** Filters groups *after* grouping and aggregation. It can use aggregate functions.

12. **Q: What are Aggregate Functions in SQL? List some.**
    - **A:** Aggregate functions perform calculations on a set of rows and return a single summary value. Examples: COUNT(), SUM(), AVG(), MIN(), MAX().
13. **Q: What is the GROUP BY clause used for?**
    - **A:** The GROUP BY clause is used in conjunction with aggregate functions to group rows that have the same values in specified columns into a summary row.
14. **Q: What is a Subquery (or Nested Query)?**
    - **A:** A subquery is a query embedded within another SQL query. It can be used in SELECT, INSERT, UPDATE, or DELETE statements, or within WHERE, HAVING, or FROM clauses.
15. **Q: What is a VIEW in SQL?**
    - **A:** A VIEW is a virtual table based on the result-set of a SQL query. It does not store data itself but derives it from other tables. Views simplify complex queries, provide security, and customize data presentation.
16. **Q: What is an INDEX in SQL and why is it used?**
    - **A:** An INDEX is a database object that provides quick access to data in a table. It's used to speed up data retrieval operations (e.g., SELECT queries) by allowing the database system to quickly locate rows without scanning the entire table.
17. **Q: What is a UNION and UNION ALL?**
    - **A:** Both combine the result-sets of two or more SELECT statements.
        - **UNION:** Combines and removes duplicate rows.
        - **UNION ALL:** Combines and retains all duplicate rows.
    - Both require the same number of columns and compatible data types in the SELECT statements.
18. **Q: What is Normalization and Denormalization in the context of SQL database design?**
    - **A:**
        - **Normalization:** A process of organizing columns and tables to minimize data redundancy and improve data integrity, typically involving breaking down large tables into smaller, related tables.
        - **Denormalization:** The process of intentionally introducing redundancy into a database by combining tables or adding duplicate data to improve query performance, often at the expense of increased data redundancy and potential for anomalies.