Database Management Systems (310241)
Viva Questions and Answers

Note on Vivas

A viva isn't just about correct answers; it's about *how* you answer. Be confident, clear, and concise. If you don't know an answer, it's better to say "I'm not sure, but I think it's related to..." than to guess wildly. Good luck!

Contents

1 Unit I: Introduction to DBMS and ER Model

1. **What is a DBMS?**
   A Database Management System (DBMS) is software that allows for the creation, retrieval, updating, and management of a database. It ensures data is organized, consistent, and secure.

2. **What are the disadvantages of a File-Based System?**
   Data redundancy (duplication), data inconsistency, difficulty in accessing data, lack of security, and integrity problems.

3. **What is Data Independence?**
   It's the ability to modify a schema definition in one level without affecting the schema definition in the next higher level.

4. **What are the two types of Data Independence?**

- **Physical Data Independence:** Ability to change the physical schema (e.g., storage structure, indexing) without changing the logical schema.
- **Logical Data Independence:** Ability to change the logical schema (e.g., add/remove attributes from tables) without changing the external schema or application programs.

5. **What is a Data Model?**
   A data model is a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.
6. **Name different types of data models.**
   Hierarchical Model, Network Model, Relational Model, Entity-Relationship (ER) Model, and Object-Oriented Model.
7. **What is an ER Model?**
   The Entity-Relationship (ER) Model is a high-level conceptual data model that describes data as entities, attributes, and relationships. It's used for database design.
8. **What is an Entity?**
   An entity is a real-world object or concept that is distinguishable from other objects. For example: `Student`, `Course`, `Employee`.
9. **What is an Attribute?**
   An attribute is a property or characteristic of an entity. For example, for a `Student` entity, attributes could be `Roll_No`, `Name`, and `Address`.
10. **What is an Entity Set?**
    An entity set is a collection of all entities of a particular entity type at any point in time. For example, the set of all students in a university.
11. **What is a Relationship?**
    A relationship is an association among two or more entities. For example, a `Student` *enrolls in* a `Course`.
12. **What are the different types of attributes?**

    - **Simple:** Cannot be divided further (e.g., `Age`).
    - **Composite:** Can be divided into sub-parts (e.g., `Address` can be `Street`, `City`, `Zip`).
    - **Single-Valued:** Has only one value (e.g., `Roll_No`).
    - **Multi-Valued:** Can have multiple values (e.g., `Phone_Number`).
    - **Derived:** Value is calculated from other attributes (e.g., `Age` from `Date_of_Birth`).
13. **What is a Key?**
    A key is an attribute or set of attributes that uniquely identifies an entity within its entity set.
14. **What is a Super Key?**
    A Super Key is a set of one or more attributes that, taken collectively, uniquely identifies an entity in an entity set.

15. **What is a Candidate Key?**
A Candidate Key is a minimal Super Key. It's a Super Key from which no attribute can be removed without it losing its uniqueness. A table can have multiple Candidate Keys.

16. **What is a Primary Key?**
A Primary Key is one of the Candidate Keys chosen by the database designer to uniquely identify tuples in a table. It cannot have NULL values.

17. **What is a Foreign Key?**
A Foreign Key is an attribute (or set of attributes) in one table that refers to the Primary Key of another table. It's used to link tables and enforce referential integrity.

18. **What is a Weak Entity?**
A weak entity is an entity that cannot be uniquely identified by its own attributes alone. It depends on its relationship with a "strong" (or owner) entity.

19. **How is a weak entity represented in an ER diagram?**
With a double-lined rectangle. The identifying relationship is shown with a double-lined diamond.

20. **What is Cardinality in a relationship?**
Cardinality defines the number of instances of one entity that can be associated with instances of another entity. Types are: One-to-One (1:1), One-to-Many (1:M), Many-to-One (M:1), and Many-to-Many (M:M).

21. **How do you convert a 1:M relationship to tables?**
The Primary Key of the "One" side entity is added as a Foreign Key to the table of the "Many" side entity.

22. **How do you convert an M:M relationship to tables?**
You create a new, separate table (called a junction or associative table). This new table's primary key is a composite key formed from the primary keys of the two original entities.

23. **What is EER?**
Extended Entity-Relationship (EER) model includes all concepts of the ER model plus advanced features like specialization, generalization, and aggregation.

24. **What is Specialization?**
A top-down process of defining a set of subclasses for an entity set. For example, `Employee` can be specialized into `Technician` and `Engineer`.

25. **What is Generalization?**
A bottom-up process of defining a more general entity type from a set of more specialized entity types. For example, `Car` and `Truck` can be generalized into `Vehicle`.

26. **What is the difference between 'disjoint' and 'overlapping' constraints in specialization?**

    - **Disjoint:** An entity can be a member of at most *one* of the subclasses (e.g., a person is either `Male` or `Female`, not both).

- **Overlapping:** An entity can be a member of *more than one* subclass (e.g., a `Person` can be both an `Employee` and a `Student`).

27. **What is Aggregation?**
Aggregation is an EER feature where a relationship between entities is treated as a single, higher-level entity. For example, a `Project` *uses* a `Machine`, and this entire relationship `(Project-uses-Machine)` can be *monitored by* an `Employee`.

28. **What are the different database languages?**

    - **DDL (Data Definition Language):** For defining the database schema (e.g., `CREATE`, `ALTER`).
    - **DML (Data Manipulation Language):** For manipulating data (e.g., `INSERT`, `SELECT`, `UPDATE`).
    - **DCL (Data Control Language):** For access control (e.g., `GRANT`, `REVOKE`).
    - **TCL (Transaction Control Language):** For managing transactions (e.g., `COMMIT`, `ROLLBACK`).

29. **Who is a DBA?**
A Database Administrator (DBA) is a person responsible for the design, implementation, maintenance, and repair of a database.

30. **What is a 'View' of data?**
A view is a virtual table based on the result-set of an SQL statement. It hides the complexity of the underlying tables and can be used for security.

31. **What are the three levels of data abstraction?**

    - **Physical Level:** The lowest level, describes *how* data is actually stored.
    - **Logical Level:** The middle level, describes *what* data is stored and what relationships exist.
    - **View Level:** The highest level, describes only a part of the entire database for specific users.

32. **What is an 'Instance' vs. a 'Schema'?**

    - **Schema:** The overall design or "blueprint" of the database (e.g., table names, column names, data types).
    - **Instance:** A snapshot of the data in the database at a particular moment in time.

33. **What is a composite key?**
A key that consists of two or more attributes that together uniquely identify an entity.

34. **What is a derived attribute?**
An attribute whose value can be calculated or derived from another attribute. It's shown with a dashed oval in an ER diagram.

35. **What is a multi-valued attribute?**
An attribute that can hold more than one value for a single entity. It's shown with a double oval.

36. **How do you convert a multi-valued attribute to a table?**
You create a new table that includes the primary key of the original entity (as a foreign key) and a new column for the multi-valued attribute.

37. **What is 'total participation'?**
In a relationship, total participation (shown by a double line) means that *every* entity in the entity set must participate in at least one relationship instance.

38. **What is 'partial participation'?**
It means that some entities in the entity set may not participate in the relationship.

39. **What is a recursive relationship?**
A relationship where an entity type relates to itself. For example, an `Employee` *manages* other `Employees`.

40. **What is an 'ar-ity' or 'degree' of a relationship?**
The number of entity types participating in a relationship. Unary (degree 1, recursive), Binary (degree 2, most common), Ternary (degree 3).

2 Unit II: SQL and PL/SQL

41. **What does SQL stand for?**
Structured Query Language.

42. **What are the sub-languages of SQL?**
DDL (Data Definition Language), DML (Data Manipulation Language), DCL (Data Control Language), and TCL (Transaction Control Language).

43. **Give examples of DDL commands.**
`CREATE`, `ALTER`, `DROP`, `TRUNCATE`, `RENAME`.

44. **Give examples of DML commands.**
`SELECT`, `INSERT`, `UPDATE`, `DELETE`.

45. **Give examples of DCL commands.**
`GRANT`, `REVOKE`.

46. **Give examples of TCL commands.**
`COMMIT`, `ROLLBACK`, `SAVEPOINT`.

47. **What is the difference between `DELETE` and `TRUNCATE`?**

- `DELETE`**:** Is a DML command, removes rows one by one, can be rolled back, and can have a `WHERE` clause.
- `TRUNCATE`**:** Is a DDL command, removes all rows at once (de-allocates data pages), cannot be rolled back, and is much faster.

48. **What is the difference between `DROP` and `TRUNCATE`?**

    - `TRUNCATE`: Removes all rows from a table, but the table structure remains.
    - `DROP`: Removes the entire table, including its structure, from the database.

49. **What is a SQL `JOIN`?**
    A `JOIN` clause is used to combine rows from two or more tables based on a related column between them.

50. **What are the different types of SQL `JOIN`s?**

    - **INNER JOIN:** Returns records that have matching values in both tables.
    - **LEFT (OUTER) JOIN:** Returns all records from the left table, and the matched records from the right table.
    - **RIGHT (OUTER) JOIN:** Returns all records from the right table, and the matched records from the left table.
    - **FULL (OUTER) JOIN:** Returns all records when there is a match in either the left or right table.

51. **What is a Self-Join?**
    A self-join is a regular join, but the table is joined with itself. It's used to query hierarchical data (e.g., finding an employee's manager).

52. **What is a Cross-Join?**
    A cross-join (or Cartesian product) returns all possible combinations of rows from both tables.

53. **What is the `WHERE` clause used for?**
    The `WHERE` clause is used to filter records. It is used to extract only those records that fulfill a specified condition.

54. **What is the `HAVING` clause used for?**
    The `HAVING` clause is used to filter groups. It is applied *after* the `GROUP BY` clause, whereas `WHERE` is applied *before*.

55. **What is the difference between `WHERE` and `HAVING`?**
    `WHERE` filters rows *before* they are grouped. `HAVING` filters groups *after* they are grouped. `WHERE` cannot be used with aggregate functions, but `HAVING` can.

56. **What are SQL Aggregate Functions?**
    Functions that operate on a set of rows and return a single value. Common examples are `COUNT()`, `SUM()`, `AVG()`, `MAX()`, and `MIN()`.

57. **What is the `GROUP BY` clause?**
    The `GROUP BY` clause groups rows that have the same values in specified columns into summary rows. It's almost always used with aggregate functions.

58. **What is a Subquery (or Nested Query)?**
    A subquery is a query (a `SELECT` statement) that is nested inside another query (e.g., inside a `WHERE`, `FROM`, or `HAVING` clause).

59. **What is a correlated subquery?**
A subquery that is executed once for each row processed by the outer query. It depends on the outer query for its values and is generally slower than a non-correlated subquery.

60. **What is a `VIEW`?**
A `VIEW` is a virtual table based on the result set of an SQL query.

61. **What are the advantages of using a View?**

    - **Security:** Restrict users to see only certain columns or rows.
    - **Simplicity:** Hide the complexity of joins and complex queries.
    - **Consistency:** Present a consistent view of data even if the underlying table structures change.

62. **Can you `INSERT` data into a `VIEW`?**
It's possible, but only if the view is based on a single table and follows several rules (e.g., must include the primary key, cannot have `GROUP BY`, etc.). It's generally not recommended.

63. **What is an `INDEX` in SQL?**
An index is a special data structure used to speed up the retrieval of rows from a table. It works like an index in a book.

64. **What is the disadvantage of an `INDEX`?**
Indexes speed up `SELECT` queries but can slow down `INSERT`, `UPDATE`, and `DELETE` operations because the index must also be updated. They also take up extra storage space.

65. **What is a `SEQUENCE`?**
A `SEQUENCE` is a database object that generates a sequence of unique numbers, often used to create primary key values automatically.

66. **What are the SQL Set Operators?**
`UNION`, `UNION ALL`, `INTERSECT`, and `MINUS` (or `EXCEPT`).

67. **What is the difference between `UNION` and `UNION ALL`?**

    - `UNION`**:** Combines the result sets of two queries and removes duplicate rows.
    - `UNION ALL`**:** Combines the result sets but does *not* remove duplicates. It's faster.

68. **What is `PL/SQL`?**
PL/SQL (Procedural Language/SQL) is Oracle's procedural extension to SQL. It allows you to write blocks of code, variables, loops, and error handling.

69. **What is a Stored Procedure?**
A stored procedure is a prepared block of SQL code that you can save and reuse. It's stored in the database and can be called by applications.

70. **What is a Stored Function?**
A stored function is similar to a stored procedure, but it *must* return a single value. It can be used directly within SQL queries.

71. **What is the difference between a Stored Procedure and a Function?**

- **Function:** Must return a value. Can be called from a `SELECT` statement.
- **Procedure:** Does not have to return a value. Cannot be called from a `SELECT` statement (must be `EXEC`uted).

72. **What is a `CURSOR`?**
A cursor is a pointer to a private SQL area that stores the result set of a query. It's used to process rows one by one in PL/SQL.

73. **What is a `TRIGGER`?**
A trigger is a special type of stored procedure that automatically executes (or "fires") in response to a DML event (`INSERT`, `UPDATE`, `DELETE`) on a specific table.

74. **What are the uses of Triggers?**
Enforcing complex business rules, maintaining data integrity, auditing changes (logging), or automatically updating other tables.

75. **What is an `ASSERTION`?**
An assertion is a constraint that specifies a condition that we wish the database to always satisfy. It's like a complex `CHECK` constraint that can involve multiple tables.

76. **What are Roles and Privileges?**

- **Privilege:** A permission to perform a specific action (e.g., `SELECT`, `INSERT`).
- **Role:** A named group of related privileges that can be granted to users, simplifying security management.

77. **What is the `LIKE` operator used for?**
It's used in a `WHERE` clause for pattern matching. The wildcard `%` matches zero or more characters, and _ matches exactly one character.

78. **What does `ORDER BY` do?**
Sorts the result set in ascending (`ASC`) or descending (`DESC`) order.

79. **What is the `IN` operator?**
A shorthand for multiple `OR` conditions. `WHERE Country IN ('USA', 'UK')` is the same as `WHERE Country = 'USA' OR Country = 'UK'`.

80. **What is the `BETWEEN` operator?**
Selects values within a given range. It's inclusive: `WHERE Price BETWEEN 10 AND 20` includes 10 and 20.

3 Unit III: Relational Database Design

81. **What is a Relational Model?**
A data model where data is stored in tables (called relations). Each table has rows (tuples) and columns (attributes).

82. **What is a 'Relation'?**
A table with columns and rows.

83. **What is a 'Tuple'?**
A single row in a table.

84. **What is an 'Attribute'?**
A named column of a relation.

85. **What is a 'Domain'?**
The set of all possible legal values for an attribute. For example, the domain for a 'Gender' attribute might be {'Male', 'Female', 'Other'}.

86. **What is 'Degree' of a relation?**
The number of attributes (columns) in the relation.

87. **What is 'Cardinality' of a relation?**
The number of tuples (rows) in the relation.

88. **What are Codd's Rules?**
A set of 12 rules (numbered 0-12) proposed by E.F. Codd, which define what a database must have to be considered truly "relational" (e.g., all data in tables, guaranteed access via keys, systematic treatment of NULLs).

89. **What is Relational Integrity?**
The rules that ensure the accuracy and consistency of data in a relational database.

90. **What are the main types of Relational Integrity?**

    - **Domain Integrity:** Enforces valid entries for a column (e.g., using data types, `CHECK` constraints).
    - **Entity Integrity:** No primary key attribute can be NULL.
    - **Referential Integrity:** A foreign key value must match an existing primary key value in the referenced table, or it must be NULL.

91. **What is an 'Enterprise Constraint'?**
A business rule that is specific to the application and not a general database rule. For example, "An employee's salary cannot be more than their manager's salary."

92. **What is Normalization?**
Normalization is the process of organizing data in a database to reduce data redundancy and eliminate undesirable anomalies (like insertion, update, and deletion anomalies).

93. **What is an 'Anomaly'?**
A problem that occurs when a database is not normalized.

- **Insertion Anomaly:** Inability to add new data due to a missing attribute.
- **Deletion Anomaly:** Accidental loss of data when a row is deleted.
- **Update Anomaly:** Having to update the same piece of data in multiple places, leading to inconsistency.

94. **What is a Functional Dependency (FD)?**
A functional dependency, denoted `A -> B`, means that the value of attribute `A` uniquely determines the value of attribute `B`.

95. **What is First Normal Form (1NF)?**
A relation is in 1NF if all its attributes have **atomic domains**. This means no multi-valued attributes or repeating groups.

96. **What is Second Normal Form (2NF)?**
A relation is in 2NF if it is in 1NF and **no non-prime attribute is partially dependent** on any candidate key. (A non-prime attribute is one not part of any candidate key).

97. **What is 'Partial Dependency'?**
A partial dependency exists when a non-prime attribute depends on *only a part* of a composite candidate key, not the whole key.

98. **What is Third Normal Form (3NF)?**
A relation is in 3NF if it is in 2NF and it has **no transitive dependencies**.

99. **What is 'Transitive Dependency'?**
A transitive dependency exists when a non-prime attribute depends on another non-prime attribute, which in turn depends on the primary key. (i.e., `A -> B` and `B -> C`, where `A` is the key, so `A -> C` is a transitive dependency).

100. **What is Boyce-Codd Normal Form (BCNF)?**
BCNF is a stricter version of 3NF. A relation is in BCNF if for every functional dependency `X -> Y`, `X` **must be a Super Key**.

101. **What is the difference between 3NF and BCNF?**
A table in BCNF is always in 3NF. A table in 3NF is not always in BCNF. BCNF addresses anomalies that 3NF might miss if there are multiple overlapping candidate keys.

102. **Is it always good to normalize to BCNF?**
Not always. Sometimes, decomposing to BCNF can result in the loss of a functional dependency, which might be undesirable. This is a trade-off between redundancy and dependency preservation.

103. **What is 'Decomposition'?**
The process of breaking down a single table into multiple, smaller tables to achieve normalization.

104. What are the properties of a good decomposition?

- **Lossless-Join:** When the decomposed tables are joined back together, they should produce the original table without any extra or missing rows.
- **Dependency Preservation:** All the original functional dependencies should be enforceable by checking constraints (like foreign keys) on the new, smaller tables.

105. What is an 'Atomic Domain'?
A domain where all values are indivisible. For example, a `Phone_Numbers` column holding `(123-456, 789-012)` is not atomic. It should be split.

106. What is a 'Prime Attribute'?
An attribute that is part of *any* candidate key.

107. What is a 'Non-Prime Attribute'?
An attribute that is not part of any candidate key.

108. What is the main goal of normalization?
To minimize data redundancy.

109. What is 'Denormalization'?
The process of intentionally introducing redundancy to a normalized database.

110. Why would you ever 'Denormalize' a database?
For performance. Joins are computationally expensive. By adding redundant data, we can avoid frequent, complex joins and speed up `SELECT` queries, which is common in data warehousing.

111. What is a trivial functional dependency?
An FD `A -> B` is trivial if `B` is a subset of `A`. For example, `(RollNo, Name) -> Name` is trivial.

112. How do you find the closure of an attribute set (e.g., A+)?
You start with the attributes in the set (A) and repeatedly add any attributes that can be determined by the attributes you already have, using the given FDs, until no more attributes can be added.

113. How do you find a candidate key using FDs?
Find an attribute set `X` such that its closure `X+` contains all attributes in the relation, and no subset of `X` has this property.

114. Explain 2NF with an example.
Table: `(StudentID, CourseID) -> (StudentName, CourseName)`
Here, `(StudentID, CourseID)` is the primary key.
FDs: `StudentID -> StudentName` and `CourseID -> CourseName`.
This is *not* in 2NF because `StudentName` (a non-prime attribute) depends only on `StudentID` (part of the key), which is a partial dependency.
Solution: Decompose into `(StudentID, StudentName)` and `(CourseID, CourseName)` and `(StudentID, CourseID)`.

115. **Explain 3NF with an example.**
Table: `(StudentID) -> (StudentName, ZipCode, City)`
Here, `(StudentID)` is the primary key.
FDs: `StudentID -> StudentName`, `StudentID -> ZipCode`, `ZipCode -> City`.
This is *not* in 3NF because `City` (a non-prime attribute) depends on `ZipCode` (another non-prime attribute), which is a transitive dependency.
Solution: Decompose into `(StudentID, StudentName, ZipCode)` and `(ZipCode, City)`.

116. **Is BCNF always dependency preserving?**
No. This is the main trade-off. 3NF decomposition is always lossless and dependency preserving, but BCNF decomposition is always lossless but *not* always dependency preserving.

117. **What is 4NF (Fourth Normal Form)?**
4NF deals with multi-valued dependencies. It states that for a dependency `A ->> B`, `A` must be a superkey.

118. **What is a multi-valued dependency (MVD)?**
An MVD `A ->> B` means that the value of `A` determines a *set* of values for `B`, independent of any other attributes.

119. **What is 5NF (Fifth Normal Form)?**
5NF deals with join dependencies. It's designed to eliminate redundancy that can occur when a table is split into three or more tables and then re-joined.

120. **Which normal form is considered sufficient for most database designs?**
3NF or BCNF are generally considered the standard for a well-designed relational database.

4 Unit IV: Database Transaction Management

121. **What is a Transaction?**
A transaction is a single logical unit of work, which consists of one or more SQL statements. It must execute completely or not at all.

122. **What are the ACID properties of a transaction?**

- **Atomicity:** The "all or nothing" rule. A transaction either completes entirely (commits) or has no effect (rolls back).
- **Consistency:** A transaction must bring the database from one valid (consistent) state to another.
- **Isolation:** Transactions running concurrently should not interfere with each other. One transaction's intermediate results are hidden from others.
- **Durability:** Once a transaction is committed, its changes are permanent and will survive any subsequent system failures (e.g., power loss).

123. **Explain Atomicity.**
If a transaction to transfer money from account A to B involves two steps (debit A, credit B), Atomicity ensures that if the 'credit B' step fails, the 'debit A' step is also undone (rolled back).

124. **Explain Isolation.**
If Transaction 1 is calculating the total balance of all accounts, and Transaction 2 is transferring money, Isolation ensures Transaction 1 sees the state *before* or *after* Transaction 2, but not an inconsistent state in the middle.

125. **What are the states of a Transaction?**

- **Active:** The initial state; transaction is executing.
- **Partially Committed:** After the last statement has executed, but before changes are made permanent.
- **Committed:** The transaction has completed successfully and changes are durable.
- **Failed:** The transaction cannot proceed normally (e.g., due to an error).
- **Aborted:** The transaction has been rolled back and the database is restored to its state before the transaction began.

126. **What is a 'Schedule'?**
A schedule (or history) is the sequence of operations (read, write, commit, abort) from one or more transactions as they are executed by the DBMS.

127. **What is a 'Serial Schedule'?**
A schedule where all operations of one transaction are executed consecutively, without any interleaving from other transactions. It's guaranteed to be consistent, but slow.

128. **What is a 'Serializable Schedule'?**
A non-serial schedule that is *equivalent* to some serial schedule. This is the goal of concurrency control, as it allows for interleaving (for performance) while guaranteeing consistency.

129. **What is 'Serializability'?**
It's the property of a schedule that ensures it is equivalent to a serial schedule.

130. **What are the two types of Serializability?**

- **Conflict Serializability:** A schedule is conflict serializable if it can be transformed into a serial schedule by swapping non-conflicting operations.
- **View Serializability:** A more relaxed form. A schedule is view serializable if it is "view equivalent" to a serial schedule.

131. **When do two operations 'conflict'?**
Two operations conflict if they belong to different transactions, access the same data item, and at least one of them is a `Write` operation.

- Read-Write (Conflict)

- Write-Read (Conflict)
- Write-Write (Conflict)
- Read-Read (No Conflict)

132. What are the problems caused by lack of concurrency control?

- **Lost Update:** Two transactions update the same item, and one update is overwritten by the other.
- **Dirty Read (Write-Read):** A transaction reads data that has been written by another transaction that has not yet committed.
- **Unrepeatable Read (Read-Write):** A transaction reads an item, another transaction modifies it, and the first transaction reads it again and gets a different value.
- **Phantom Read:** A transaction runs a query, another transaction inserts new rows that match the query, and the first transaction re-runs the query and gets *new* (phantom) rows.

133. What is Concurrency Control?
The process of managing simultaneous operations on the database without them interfering with each other.

134. What are the main Concurrency Control techniques?

- Lock-Based Protocols
- Timestamp-Based Protocols
- Optimistic Protocols (Validation-Based)

135. What is a 'Lock'?
A lock is a mechanism to control access to a data item. A transaction must acquire a lock on an item before it can operate on it.

136. What are the types of locks?

- **Shared (S) Lock / Read Lock:** Multiple transactions can hold a shared lock on an item. Used for reading.
- **Exclusive (X) Lock / Write Lock:** Only one transaction can hold an exclusive lock on an item. Used for writing.

137. What is the Two-Phase Locking (2PL) protocol?
A protocol that ensures serializability. It has two phases:

(a) **Growing Phase:** The transaction can acquire new locks but cannot release any.

(b) **Shrinking Phase:** The transaction can release locks but cannot acquire any new ones.

138. **Does 2PL prevent deadlocks?**
No, standard 2PL ensures serializability but can still lead to deadlocks.

139. **What is 'Strict 2PL'?**
A stricter version of 2PL where a transaction must hold all its *exclusive* locks until it commits or aborts. This prevents dirty reads and avoids "cascading aborts".

140. **What is 'Rigorous 2PL'?**
The strictest form. It holds *all* locks (shared and exclusive) until the transaction commits or aborts.

141. **What is a 'Cascading Abort' (or Cascading Rollback)?**
A situation where the failure of one transaction (T1) causes a second transaction (T2) to also be aborted, because T2 read data written by T1 (a dirty read).

142. **What is a 'Recoverable Schedule'?**
A schedule where a transaction T2 that reads data from T1 is only allowed to commit *after* T1 has committed. This prevents dirty reads.

143. **What is a 'Deadlock'?**
A situation where two or more transactions are in a circular wait, each waiting for a lock held by the next transaction in the circle.

144. **How do you handle deadlocks?**

- **Deadlock Prevention:** Prevent the circular wait from ever happening (e.g., by acquiring all locks at once, or ordering lock requests).
- **Deadlock Detection:** Periodically check for cycles in a "wait-for graph". If a cycle is found, one transaction (the "victim") is aborted and restarted.

145. **What is 'Timestamp-Based' concurrency control?**
Each transaction is given a unique timestamp when it starts. The DBMS checks the timestamp of transactions and data items to resolve conflicts, ensuring that operations are processed in a serializable order (oldest-first).

146. **What is Database Recovery?**
The process of restoring the database to its last consistent state after a failure (e.g., system crash, disk failure).

147. **What is 'Log-Based Recovery'?**
A recovery method that uses a 'log' file. The log contains a record of all transaction operations, such as `[T1, start]`, `[T1, write, A, 10, 20]`, `[T1, commit]`.

148. **What is a 'Checkpoint'?**
A checkpoint is a point in time where the DBMS flushes all modified log records and data from memory (buffers) to disk. This speeds up recovery, as the system only needs to replay the log *after* the last checkpoint.

149. **What is 'Deferred Database Modification'?**
A recovery technique where write operations are not applied to the database on disk until the transaction commits. If it aborts, no "undo" is needed.

150. **What is 'Immediate Database Modification'?**
A technique where write operations are applied to the database on disk *before* the transaction commits. This requires an "undo" operation if the transaction later aborts.

151. **What is 'Shadow Paging'?**
A recovery technique that maintains two "page tables": a current one and a "shadow" one. When a write occurs, a copy of the page is made, modified, and the *current* page table is updated. If the transaction aborts, the shadow table (which points to the original, unmodified data) is simply reinstated.

152. **What is ACID's 'Durability' usually implemented with?**
Write-Ahead Logging (WAL). The rule is: the log record for a change must be written to stable storage (disk) *before* the change itself is written to the database on disk.

153. **What is a 'View Serializable' schedule?**
A schedule is view serializable if it is "view equivalent" to some serial schedule. This means:

    (a) They both perform the same initial reads.
    (b) They both perform the same final writes.
    (c) They both have the same "read-from" relationships.

154. **Is Conflict Serializability stricter or View Serializability?**
Conflict Serializability is *stricter*. All conflict-serializable schedules are view-serializable, but not all view-serializable schedules are conflict-serializable.

155. **What is the 'Timestamp Ordering' protocol?**
A concurrency protocol where if a transaction T1 tries to access data that was already accessed by a *younger* transaction T2, T1 is aborted and restarted with a new, later timestamp.

5 Unit V: NoSQL Databases

146. **What is a Distributed Database System?**
A database system where the data is stored across multiple physical locations (sites or nodes), connected by a network.

147. **What are the advantages of a Distributed Database?**

    - **Scalability:** Can add more nodes to handle more data (horizontal scaling).
    - **Availability/Reliability:** If one node fails, the system can continue operating using other nodes.
    - **Performance:** Data can be stored closer to the users who need it, reducing network-latency.

148. **What are the disadvantages of a Distributed Database?**
Increased complexity in query processing, transaction management (distributed commits), and maintaining data consistency.

149. **What is the CAP Theorem?**
A fundamental theorem for distributed systems. It states that a distributed data store can only provide *at most two* of the following three guarantees:

- **Consistency (C):** All nodes see the same data at the same time.
- **Availability (A):** Every request receives a non-error response, even if nodes are down.
- **Partition Tolerance (P):** The system continues to operate despite network failures ("partitions") that split the system into non-communicating parts.

150. **Why must a distributed system choose 'P'?**
In a real-world distributed system, network failures *will* happen. Therefore, the system *must* be partition tolerant. The real trade-off is between Consistency and Availability (C or A).

151. **What is a 'CP' system? (e.g., MongoDB, Redis)**
A system that chooses **Consistency** and **Partition Tolerance**. If a network partition occurs, the system will become *unavailable* (e.g., stop accepting writes) to ensure that consistency is maintained.

152. **What is an 'AP' system? (e.g., Cassandra, CouchDB)**
A system that chooses **Availability** and **Partition Tolerance**. If a partition occurs, the system remains *available*, but some nodes may return stale or conflicting data. It aims for "eventual consistency."

153. **What is Unstructured Data?**
Data that does not have a pre-defined data model or is not organized in a pre-defined manner. Examples: text in an email, images, videos, social media posts.

154. **What is Semi-Structured Data?**
Data that doesn't conform to a rigid tabular model, but contains tags or markers to separate semantic elements. Examples: JSON, XML.

155. **What is NoSQL?**
"Not Only SQL." It's a class of databases that are non-relational, distributed, open-source, and horizontally scalable. They are designed to handle large volumes of unstructured and semi-structured data.

156. **Why do we need NoSQL?**
Relational databases (SQL) struggle with the "3 V's" of Big Data:

- **Volume:** Massive amounts of data.
- **Velocity:** Data being created at high speed.
- **Variety:** Handling unstructured and semi-structured data.

NoSQL databases are built to scale out horizontally and are more flexible.

157. **What are the BASE Properties?**
An alternative to ACID, common in NoSQL databases:

- **B**asically **A**vailable: The system guarantees availability (an 'A' in CAP).

- **S**oft State: The state of the system may change over time, even without input.
- **E**ventually Consistent: The system will *eventually* become consistent once all inputs stop, but it's not guaranteed to be consistent at every moment.

158. Compare ACID vs. BASE.

- **ACID:** Pessimistic, prioritizes Consistency. (e.g., a banking transaction).
- **BASE:** Optimistic, prioritizes Availability. (e.g., a social media "like" count, where it's okay if it's not 100% accurate for a few seconds).

159. What are the 4 main types of NoSQL Databases?

- **Key-Value Stores**
- **Document Stores**
- **Graph Databases**
- **Wide-Column (or Column-Family) Stores**

160. What is a Key-Value Store?
The simplest NoSQL type. Data is stored as a collection of key-value pairs. Very fast for simple lookups.

- **Example:** Redis, Amazon DynamoDB.

161. What is a Document Store?
Stores data in "documents" (often JSON or BSON). Each document is self-contained and can have a different, flexible schema.

- **Example:** MongoDB, CouchDB.

162. What is a Graph Database?
Designed to store and query data as nodes and edges (relationships). Excellent for highly connected data.

- **Example:** Neo4j, Amazon Neptune.

163. What is a Wide-Column Store?
Stores data in tables, rows, and *dynamic* columns. Columns are grouped into "column families." Excellent for large-scale queries over massive datasets.

- **Example:** Apache Cassandra, Google Bigtable.

164. What is MongoDB?
A popular open-source, document-oriented NoSQL database.

165. What is a 'Collection' in MongoDB?
A group of MongoDB documents. It's the equivalent of a 'table' in an RDBMS.

166. What is a 'Document' in MongoDB?
A set of key-value pairs, stored in a BSON (binary JSON) format. It's the equivalent of a 'row' in an RDBMS, but much more flexible.

167. What are CRUD operations?
Create, Read, Update, Delete. These are the four basic functions of persistent storage.

168. How do you perform CRUD in MongoDB?

- **Create:** `db.collection.insertOne()` or `insertMany()`
- **Read:** `db.collection.find()` or `findOne()`
- **Update:** `db.collection.updateOne()` or `updateMany()`
- **Delete:** `db.collection.deleteOne()` or `deleteMany()`

169. What is Indexing in MongoDB?
The same as in SQL: it creates a special data structure to speed up `find()` queries. MongoDB automatically creates an index on the `_id` field.

170. What is 'Aggregation' in MongoDB?
A framework for performing data processing and analysis. It processes documents in a "pipeline" of stages (e.g., `$match`, `$group`, `$sort`) to return a computed result.

171. What is 'MapReduce'?
An older, more complex aggregation model. It involves a `map` function that processes and "emits" key-value pairs, and a `reduce` function that aggregates the emitted data. The Aggregation Pipeline is now preferred.

172. What is 'Replication' in MongoDB?
The process of synchronizing data across multiple servers. MongoDB uses a "replica set" (one Primary node, multiple Secondary nodes) to provide high availability and redundancy.

173. What is 'Sharding' in MongoDB?
The process of distributing data across multiple servers (or "shards"). It's MongoDB's solution for horizontal scaling (handling massive data volume).

174. What is the difference between Replication and Sharding?

- **Replication:** *Copies* the same data to multiple servers (for high availability).
- **Sharding:** *Splits* the data into parts, with each part on a different server (for high volume/scalability).

A production system uses both.

175. Why is MongoDB "schema-less"?
It's more accurately "schema-flexible." It means documents in the same collection do not need to have the same set of fields or structure.

176. When would you use RDBMS (SQL) over NoSQL?

- When you need strong ACID transactional guarantees (e.g., financial systems).
- When your data is highly structured and relational.

- When your data schema is stable and not expected to change often.

177. When would you use NoSQL over RDBMS?

- When you need to store large volumes of unstructured or semi-structured data.
- When you need very high read/write speed.
- When you need to scale horizontally (add more servers) easily.

178. What is the `_id` field in MongoDB?
A unique primary key for every document in a collection. If you don't provide one, MongoDB automatically generates a unique 12-byte `ObjectId`.

179. Can you have joins in MongoDB?
Yes, using the `$lookup` operator in the aggregation pipeline. It performs a "left outer join" with another collection. However, joins are less common and less performant than in SQL.

180. What is BSON?
Binary JSON. It's a binary-encoded serialization format for JSON-like documents. It supports more data types than JSON (e.g., `Date`, `ObjectId`) and is optimized for speed and storage.

6 Unit VI: Advances in Databases

181. What is an Active Database?
A database that includes event-driven rules (e.g., triggers). It can react automatically to events or changes in data, without needing an external application to initiate the action.

182. What is a Deductive Database?
A database that can make logical deductions (inferences) based on facts and rules stored in the database. It combines database technology with logic programming.

183. What is a Main Memory Database (MMDB)?
An In-Memory Database (IMDB). It's a database that stores all its data in the main memory (RAM) instead of on disk.

184. What is the primary advantage of a Main Memory Database?
Extreme speed. Accessing data in RAM is thousands of times faster than accessing it from a hard disk (HDD) or even a solid-state drive (SSD).

185. What is the primary disadvantage of a Main Memory Database?

- **Volatility:** RAM is volatile, so all data is lost on power failure (it must be backed by non-volatile storage and logs).
- **Cost:** RAM is much more expensive per-GB than disk storage.

186. What is Semi-Structured Data?
Data that has some organizational properties (like tags or markers) but doesn't fit the rigid structure of a relational table.

187. What is XML?
e**X**tensible **M**arkup **L**anguage. It's a text-based format for representing semi-structured data using custom tags. It's human-readable and machine-readable.

188. What is JSON?
JavaScript **O**bject **N**otation. It's a lightweight, text-based data interchange format based on key-value pairs and arrays.

189. What are the key differences between JSON and XML?

- **Syntax:** JSON is less verbose and cleaner (uses {} and `[]`). XML uses opening and closing tags `<tag>...</tag>`.
- **Data Types:** JSON has built-in types (string, number, boolean, array, object). XML is just text; data types must be defined by a schema (XSD).
- **Parsing:** JSON is generally faster for machines to parse.
- **Usage:** JSON is the de-facto standard for modern web APIs. XML is still common in enterprise systems, configuration files, and document-oriented applications.

190. What is an Object-Relational Database System (ORDBMS)?
A database system that tries to "bridge the gap" between relational databases and object-oriented programming. It supports objects, classes, and inheritance directly in the database.

- **Example:** PostgreSQL.

191. What is Table Inheritance (as seen in PostgreSQL)?
A feature where a table can "inherit" all the columns from a parent table. A query on the parent table can optionally include all rows from its child tables.

192. What is Object-Relational Mapping (ORM)?
ORM is a **programming technique**, not a database type. It's a library (e.g., SQLAlchemy for Python, TypeORM for Node.js) that "maps" objects in an object-oriented programming language (like a `User` class) to tables in a relational database (like a `users` table).

193. What is the advantage of using an ORM?
It allows developers to work with database tables as if they were native programming objects, without having to write raw SQL queries.

194. What is the "impedance mismatch"?
The conceptual difference between the object-oriented model (with classes, inheritance, and complex objects) and the relational model (with flat tables, rows, and columns). ORMs try to solve this.

195. What is Spatial Data?
Data that represents geographic or geometric information about objects in space.

196. What are the two main types of Spatial Data?

- **Geometric Data:** Represents shapes like points, lines, and polygons in a 2D or 3D coordinate system.
- **Geographic Data:** A specific type of geometric data that is mapped to a location on Earth (e.g., using latitude and longitude).

197. Give an example of a Spatial Database.

- **PostGIS** (an extension for PostgreSQL).
- MySQL and SQL Server also have spatial data types and functions.

198. What kind of query would you run on a spatial database?

- "Find all coffee shops within 1 kilometer of my current location." (A proximity query).
- "Find all parks that intersect with this neighborhood boundary." (An intersection query).

199. What is a 'nested data type'?
A data type where a column in a table can itself contain complex values, like another table, an array, or a JSON object. This is a move away from 1NF.

200. How does JSON support nested data types?
A JSON object's value can be another JSON object or an array of JSON objects, allowing for infinite nesting.

201. Why are nested data types like JSON useful?
They allow you to store related data together in a single document or row, which can be faster to retrieve than joining multiple tables.

202. What is an R-tree?
A special type of index used by spatial databases to efficiently query multi-dimensional data (like 2D or 3D coordinates).

203. What is a Deductive Database rule?
A rule in the form `head :- body`. It means "If the *body* is true, then the *head* is also true." For example: `ancestor(X, Y) :- parent(X, Y).`

204. What is the difference between an Active DB and a Deductive DB?

- **Active DB:** Reacts to *changes* in data (events).
- **Deductive DB:** Answers queries by *inferring* new facts from existing facts and rules.

205. What is table inheritance useful for?
Modeling "is-a" relationships. For example, you could have a `city` table and a `capital_city` table that inherits from `city` (and adds a `state_govt` column).

7 Bonus "Deep Dive" and Comparison Questions

206. (I) What is the main job of the Database Manager?
It's the central software component of the DBMS. It handles query processing, storage management, transaction management, and provides an interface between the low-level data and the application programs.

207. (I/III) How do you represent a weak entity in a relational table schema?
The table for the weak entity *must* include the primary key of its "owner" strong entity as a foreign key. The primary key of the weak entity's table is a composite key, made of its own partial key (discriminator) and the foreign key from the owner.

208. (II) What is the difference between a 'clustered' and 'non-clustered' index?

- **Clustered Index:** Physically re-orders the rows in the table based on the indexed column. A table can only have *one* clustered index.
- **Non-Clustered Index:** Creates a separate data structure (like a B-Tree) that holds pointers to the actual data rows. A table can have many non-clustered indexes.

209. (II) If a query is slow, what is the *first* thing you should check?
The query's execution plan (e.g., using `EXPLAIN` in SQL). This will show you if the query is using indexes effectively or if it's doing a full table scan.

210. (III) Why is BCNF considered "stronger" than 3NF?
Because *every* BCNF relation is also in 3NF, but not every 3NF relation is in BCNF. BCNF eliminates all functional-dependency-based redundancy, while 3NF permits some redundancy if a non-prime attribute depends on a key that isn't the primary key.

211. (IV) What is the 'Phantom Read' problem and how is it solved?
A phantom read happens when a transaction re-runs a query and finds *new rows* that were inserted by another committed transaction. This is solved by a higher isolation level called "Serializable," which often uses "predicate locking" (locking a `WHERE` clause condition, not just the rows).

212. (IV) What is a 'Dirty Read' and what isolation level prevents it?
A dirty read is reading data that has been modified by another transaction but *not yet committed*. The "Read Committed" isolation level prevents this by ensuring a transaction can only read data that has been durably saved.

213. (V) What is a 'column-family' in a Wide-Column store?
It's a group of related columns that are often accessed together. In Cassandra, the column family is the basic unit of storage and is roughly equivalent to a table.

214. (V) If you were building a social network, what database type would you use for the "friends" list?
A **Graph Database** (like Neo4j). It is optimized for "many-to-many" relationship queries like "find all friends-of-friends."

215. (V) If you were building a comments section for a blog, what type might you use?
A **Document Database** (like MongoDB). You could store each blog post as a document and embed all its comments as a nested array within that same document, making it very fast to retrieve a post and all its comments in a single read.

216. (VI) What is the difference between ORDBMS and an ORM?

- **ORDBMS:** Is the **database system itself** (e.g., PostgreSQL) which has object-oriented features built-in.
- **ORM:** Is a **library in your application code** (e.g., SQLAlchemy) that *translates* between your application's objects and a *relational* database's tables.

217. (VI) What is a 'Semantic Database'?
A database built on a "semantic network" or "ontology." It stores data in terms of *meaning* and relationships, often as triples (Subject, Predicate, Object). It's the foundation for the "Semantic Web."

218. (III/V) How does normalization in RDBMS relate to document design in NoSQL?
They are often opposites. RDBMS (SQL) *normalizes* data by splitting it into many tables (to reduce redundancy). NoSQL (e.g., MongoDB) often *denormalizes* data by embedding related information (like an `address` object) inside a main document (like a `user` document) to speed up reads.

219. (II) What is the `HAVING` clause and why is it different from `WHERE`?
`WHERE` filters rows *before* they are grouped by `GROUP BY`. `HAVING` filters *groups* of rows *after* they have been aggregated by `GROUP BY`. You use `HAVING` to filter on an aggregate function, e.g., `HAVING COUNT(*) > 10`.

220. (IV) What is a 'Wait-for Graph'?
A directed graph used for deadlock detection. A node is created for each transaction. An edge `T1 -> T2` is added if T1 is waiting for a lock held by T2. A **cycle** in this graph indicates a deadlock.

221. (I) What is the difference between a 'Database' and a 'Database System (DBMS)'?

- **Database:** The collection of data itself, stored in files.
- **DBMS:** The *software* that manages and interacts with the database (e.g., MySQL, Oracle, MongoDB).

222. (II) What is SQL Injection?
A common web attack where an attacker inserts malicious SQL code into a query (e.g., via a web form) to bypass security and access or corrupt data.

223. (II) How do you prevent SQL Injection?
By *never* concatenating user input directly into a query string. Always use **Parameterized Queries** (or Prepared Statements), where the user input is sent as a separate parameter and is never executed as code.

224. (III) What is a 'Lossless-Join Decomposition'?
A decomposition of a table `R` into `R1` and `R2` is lossless if, when you `NATURAL JOIN` `R1` and `R2` back together, you get the exact same table `R` you started with, with no spurious (extra) or missing tuples.

225. (V) What is 'Eventual Consistency'?
A consistency model used by AP systems (like Cassandra). It guarantees that *if* no new updates are made to an item, *eventually* all replicas of that item will converge to the same value.

226. (I) Who is E.F. Codd?
An IBM computer scientist who, in 1970, published the paper "A Relational Model of Data for Large Shared Data Banks," which was the foundation for the entire relational database industry.

227. (II) What is `DCL` (Data Control Language)?
It's the set of SQL commands used to control access and permissions. The two main commands are `GRANT` (to give a user a privilege) and `REVOKE` (to take it away).

228. (IV) What is 'Optimistic' Concurrency Control?
A technique that assumes conflicts are rare. Transactions are allowed to execute without acquiring locks. Before committing, the transaction *validates* to see if any other transaction has modified the data it used. If a conflict is found, the transaction is rolled back and restarted.

229. (V) What is a 'Replica Set' in MongoDB?
The mechanism for providing replication and high availability. It's a group of `mongod` processes, one of which is the **Primary** (accepts all writes) and the others are **Secondaries** (replicate the Primary's data). If the Primary fails, the Secondaries elect a new Primary.

230. (VI) What is a 'Geofence'?
A virtual perimeter for a real-world geographic area. Spatial databases can be used to send alerts when a device (like a phone or truck) enters or leaves a geofenced area.

231. (II) What is a `CURSOR` and why would you use it?
A cursor is a database object used to iterate through the result set of a query, one row at a time. You use it in PL/SQL or stored procedures when you need to perform complex, row-by-row processing that can't be done in a single SQL statement.

232. (III) Why is 'referential integrity' so important?
It prevents "orphan records." By using foreign keys, it ensures that a row in one table (e.g., an `order` row) cannot reference a `customer_id` that no longer exists in the `customers` table. It keeps the data relationships consistent.

233. **(I) What is the difference between a 'Super Key' and a 'Candidate Key'?**
A Super Key is *any* set of attributes that uniquely identifies a row. A Candidate Key is a *minimal* Super Key, meaning you can't remove any attribute from it, and it still remains a unique identifier.

234. **(II) What is the `COALESCE` function?**
A standard SQL function that returns the first non-NULL value in a list of arguments. It's useful for providing a default value, e.g., `COALESCE(MiddleName, 'N/A')`.

235. **(IV) What is the 'Durability' in ACID, and how is it achieved?**
Durability ensures that once a `COMMIT` is issued, the changes are permanent, even if the system crashes. It's achieved using **Write-Ahead Logging (WAL)**, where all changes are written to a transaction log on disk *before* they are applied to the main database files.

236. **(V) What is a 'Column-Family' store also called?**
A Wide-Column Store.

237. **(V) Give an example of an application for each NoSQL type.**

   - **Key-Value:** Caching (e.g., user sessions).
   - **Document:** Content management, user profiles (e.g., a blog post with comments).
   - **Graph:** Social networks, recommendation engines.
   - **Wide-Column:** IoT sensor data, time-series data.

238. **(VI) What is a 'Data Warehouse'?**
A large, central repository of data that is aggregated from one or more sources. It's designed for business intelligence, reporting, and analysis (OLAP), not for real-time transactions (OLTP).

239. **(VI) What is 'OLTP' vs. 'OLAP'?**

   - **OLTP (Online Transaction Processing):** Manages day-to-day transactions (e.g., an e-commerce order). Databases are highly normalized and optimized for fast `INSERT/UPDATE`.
   - **OLAP (Online Analytical Processing):** Manages complex analysis and reporting (e.g., "sales trends by region over 5 years"). Databases (warehouses) are denormalized (e.g., Star Schema) and optimized for complex `SELECT` queries.

240. **(I) What is a 'Ternary Relationship' in an ER diagram?**
A relationship that involves three distinct entity types. For example, a `Doctor` *prescribes* a `Drug` to a `Patient`. This single event links all three.

241. **(II) What is the difference between `CHAR` and `VARCHAR`?**

   - `CHAR(10)`: Fixed-length. It *always* reserves 10 characters of space, even if you only store "Hi".

- `VARCHAR(10)`: Variable-length. It stores "Hi" using only the space it needs (plus a small overhead).

Use `CHAR` for fixed-length data (e.g., state abbreviations like 'CA', 'NY'). Use `VARCHAR` for variable-length data (e.g., names).

242. (III) What is a 'functional dependency' (FD) in simple terms?
If you know the value of attribute `A`, you can *uniquely* find the value of attribute `B`. `A` determines `B`. For example, `StudentID -¿ StudentName`.

243. (IV) What is a 'stalemate'?
Another term for 'deadlock'.

244. (V) Is MongoDB 'CP' or 'AP'?
By default, MongoDB is a **CP** system. In a network partition, it will make the non-primary partition *unavailable* for writes to ensure that all data remains *consistent.*

245. (V) Is Cassandra 'CP' or 'AP'?
Cassandra is an **AP** system. It prioritizes *availability.* In a partition, all nodes remain available for reads and writes. It resolves conflicts later, aiming for eventual consistency.

246. (VI) Is a JSON object a good data type for a relational database?
It can be, but it's a trade-off. Modern SQL databases (like PostgreSQL) have excellent `JSONB` (binary JSON) types. It's useful for storing flexible, semi-structured data. However, it's harder to query and index the data *inside* the JSON blob, and it violates 1NF.

247. (II) What is the `ANY` and `ALL` operator?
They are used with subqueries.

- `> ANY (subquery)`: True if the value is greater than *at least one* value returned by the subquery.
- `> ALL (subquery)`: True if the value is greater than *all* values returned by the subquery.

248. (IV) What is the simplest way to prevent deadlocks?
Lock Ordering. If all transactions must acquire locks in the same predefined order (e.g., always lock `Table A` before `Table B`), a circular wait is impossible.

249. (I) How do you map a 1:1 relationship to tables?
You have two main options:

(a) **Merge:** Combine the two entities into a single table.
(b) **Foreign Key:** Add the primary key of one table as a foreign key in the other (and possibly add a `UNIQUE` constraint).

250. (III) What is the final goal of normalization?
To create a database schema where each fact is stored in exactly one place, minimizing redundancy and maximizing data integrity.