VIVA QUESTIONS

1. Define Data.

2. Define Information.

3. Define Database.

4. Define DBMS.

5. What do you mean by processed data?

6. What do you mean by data management?

7. Which are the actions that are performed on the database?

8. Mention the different types of DBMS.

9. Define Data model.

10. Mention the different types of Data models.

11. Why database approach is advantageous than the file system approach?

12. Who is called as the father of RDBMS?

13. What do you mean by redundant data?

14. What do you mean by Data duplication?

15. Mention the different relational algebra operations.

16. Mention the different User interfaces provided by the database system.

17. Mention the different languages provided by the database system

18. What is the difference between select operation in relational algebra and in SQL?

19. What is the difference between JOIN and Cartesian product?

20. Mention the different types of Join operations.

21. What is the difference between EQUIJOIN and NATURAL JOIN?

22. What is the difference between OUTER JOIN and JOIN.?

23. What is the difference between OUTER UNION and UNION?

24. What do you mean by Union Compatibility.?

25. What do you mean by Type Compatibility?

26. Mention the different types of relational constraints.

27. Mention the different types of structural constraints

28. What do you mean by cardinality?

29. What do you mean by cardinality ratio?

30. What do you mean by degree of a relation?

31. What do you mean by entity integrity constraint?

32. What do you mean by referential integrity constraint?

33. What do you mean by NULL constraint?

34. What do you mean by unique constraint?

35. What do you mean by Check constraint?

36. Define functional dependency.

37. Define normalization.

38. Define normal form

39. Mention the different types of normal forms

40. What is the difference between 3NF and BCNF?

41. What do you mean by JOIN dependencies?

42. What do you mean by Inclusion dependencies?

43. What do you mean by Template dependencies?

44. What do you mean by Multivalued dependencies?

45. Define Project Join Normal form.

46. Define Domain Key Normal form.

47. Mention the informal guidelines for database design.

48. Define super key.

49. Define primary key.

50. Define foreign key.

51. Define unique key.

52. Define prime attribute.

53. Define trivial functional dependency.

54. When a FD is said to be fully FD?

55. Mention the different Armstrong's inference rules.

56. Why Armstrong's inference rules are said to be sound and complete?

57. Define denormalisation.

58. Define Transaction.

59. Mention the ACID properties.

60. Define schedule.

61. Is DBMS usage always advisable or some times we may depend on file base systems? Comment on the statement by describing the situation where DBMS is not a better option & file base systems is better.

62. Describe 3-level architecture of DBMS with details of languages associated at different levels plus the level of data independence.

63. How logical architecture of DBMS differs from physical architecture?

64. Create an E R diagram and relational schema to hold information about the situation in many institutions affiliated to some University, many teachers of different disciplines are teaching to many students enrolled in many courses offered by the university to the students through the institutions. Use concept of keys, aggregation, generalisation, cardinality etc. in a proper way.

65. What is the utility of relational algebra & relational calculus? Name some software's based on these concepts?

66. Comment on the statement "Set theory has contributed a lot to RDBMS" support it with the help of suitable examples.

67. "Redundancy of data is many times beneficial" Justify the statement, also describe the situation when redundancy will mess up the current data base status, at that instance of time what actions you will prefer to take.

68. In Oracle we are having variety of versions Oracle 8, Oracle 9, etc, what does the associated number mean. Again, we are having Oracle 8i, Oracle 9i etc, what does this "i" mean.

69. Describe the various file organization techniques? How a binary tree is different from B-tree and B+ tree? Under which situation we need to use B+ tree or B tree.

Prove "Any relation which is in BCNF is in 3NF, but converse is not true"

70. Which functional dependencies are to be removed to achieve respective normal form? Discuss all the normal forms up to 4NF?

71. What is the mathematical basis of SQL? The SQL statement: select * from student will perform like projection or selection? Give details in support of your answer.

VIVA ANSWERS

1. **Data**: Data refers to raw facts and figures that are devoid of context and have not been processed to derive meaning.

2. **Information**: Information is processed data that has been organized, structured, or presented in a context that gives it meaning and relevance.

3. **Database**: A database is an organized collection of structured data, typically stored electronically in a computer system, and accessible in various ways. It is designed to efficiently manage, store, retrieve, and update data.

4. **DBMS (Database Management System)**: A DBMS is a software system that provides an interface for users to interact with the database by entering, manipulating, and retrieving data. It also includes tools for managing databases, ensuring data integrity, and controlling access to the data.

5. **Processed data**: Processed data refers to information that has undergone some form of manipulation, transformation, or analysis to make it meaningful and useful for decision-making or other purposes.

6. **Data management**: Data management involves the process of acquiring, validating, storing, protecting, and processing data to ensure its accuracy, reliability, and accessibility throughout its lifecycle.

7. **Actions performed on the database**: These actions typically include querying, inserting, updating, and deleting data, as well as managing the database schema, security, and performance.

8. **Types of DBMS**:
   - Relational DBMS (RDBMS)
   - Object-oriented DBMS (OODBMS)
   - NoSQL DBMS
   - NewSQL DBMS
   - Graph DBMS

9. **Data model**: A data model is a conceptual representation of the data structures, relationships, and rules that govern the storage and manipulation of data in a database system.

10. **Types of Data models**:
    - Hierarchical data model
    - Network data model
    - Relational data model
    - Object-oriented data model
    - Entity-relationship model
    - XML data model

11. **Advantages of database approach over file system approach**: Database approach offers advantages such as data integrity, data security, data independence, efficient data access, concurrency control, and easier data manipulation compared to the file system approach.

12. **Father of RDBMS**: Dr. Edgar F. Codd is often referred to as the father of the relational database management system (RDBMS).

13. **Redundant data**: Redundant data refers to the presence of duplicate or repetitive information in a database, which can lead to inefficiency and inconsistency.

14. **Data duplication**: Data duplication occurs when the same data is stored in multiple locations within a database or across different databases.

15. **Relational algebra operations**: Common relational algebra operations include SELECT, PROJECT, JOIN, UNION, INTERSECT, and DIFFERENCE.

16. **User interfaces provided by the database system**: Examples include command-line interfaces (CLI), graphical user interfaces (GUI), web-based interfaces, and application programming interfaces (APIs).

17. **Languages provided by the database system**:
    - SQL (Structured Query Language)
    - PL/SQL (Procedural Language/SQL)
    - T-SQL (Transact-SQL)
    - DML (Data Manipulation Language)

- DDL (Data Definition Language)
- DCL (Data Control Language)

18. **Difference between select operation in relational algebra and in SQL**: In relational algebra, SELECT operation is used to retrieve rows from a relation that satisfy a given condition. In SQL, the SELECT statement is used to retrieve data from one or more tables based on specified criteria.

19. **Difference between JOIN and Cartesian product**: JOIN combines rows from two or more tables based on a related column between them, while Cartesian product combines all rows from two tables regardless of any related column.

20. **Types of Join operations**: Common types of join operations include INNER JOIN, OUTER JOIN (LEFT, RIGHT, FULL), CROSS JOIN, and NATURAL JOIN.

21. **Difference between EQUIJOIN and NATURAL JOIN**: EQUIJOIN is a type of join that combines rows from two tables based on a specified condition, while NATURAL JOIN is a type of join that combines rows based on columns with the same name and data type in both tables.

22. **Difference between OUTER JOIN and JOIN**: JOIN typically refers to an INNER JOIN, where only matching rows from both tables are included. OUTER JOIN includes all rows from at least one of the tables, even if there is no matching row in the other table.

23. **Difference between OUTER UNION and UNION**: UNION combines the results of two queries into a single result set, removing duplicates, while OUTER UNION combines the results of two queries including duplicates and fills in missing values with NULLs.

24. **Union Compatibility**: Union compatibility refers to the requirement that for two relations to be combined using UNION, they must have the same number of attributes, and the corresponding attributes must have compatible data types.

25. **Type Compatibility**: Type compatibility refers to the requirement that when combining attributes from different relations, their data types must be compatible or convertible to a common data type.

26. **Types of relational constraints**:

- Entity integrity constraint

- Referential integrity constraint
- Domain constraint

27. **Types of structural constraints**:
    - Key constraint
    - Foreign key constraint
    - Check constraint
    - Default constraint

28. **Cardinality**: Cardinality refers to the number of occurrences of one entity that is related to the number of occurrences of another entity in a relationship.

29. **Cardinality ratio**: Cardinality ratio specifies the relationship between entities in terms of their cardinalities, such as one-to-one, one-to-many, or many-to-many.

30. **Degree of a relation**: The degree of a relation refers to the number of attributes (columns) in a relation (table).

31. **Entity integrity constraint**: Entity integrity constraint ensures that each row in a table has a unique identifier, typically achieved through the use of primary keys.

32. **Referential integrity constraint**: Referential integrity constraint ensures that relationships between tables remain consistent, typically enforced through the use of foreign keys.

33. **NULL constraint**: NULL constraint specifies that a column can contain NULL values, meaning it can have missing or unknown data.

34. **Unique constraint**: Unique constraint ensures that all values in a column or combination of columns are distinct from one another, except for NULL values.

35. **Check constraint**: Check constraint specifies a condition that must be satisfied for data to be entered or updated in a column.

36. **Functional dependency**: Functional dependency is a relationship between attributes in a relation, where the value of one attribute uniquely determines the value of another attribute.

37. **Normalization**: Normalization is the process of organizing data in a database to reduce redundancy and dependency by dividing large tables into smaller tables and defining relationships between them.

38. **Normal form**: Normal form is a way of measuring the levels of normalization achieved in a database design, with each level (1NF, 2NF, 3NF, BCNF, etc.) representing a higher degree of normalization and fewer anomalies.

39. **Types of normal forms**:

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce-Codd Normal Form (BCNF)
- Fourth Normal Form (4NF)
- Fifth Normal Form (5NF)

40. **Difference between 3NF and BCNF**:

- **Third Normal Form (3NF)**: In 3NF, a relation is in second normal form (2NF) and no non-prime attribute is transitively dependent on the primary key. This means that every non-prime attribute is directly dependent on the primary key and not on any other non-prime attribute.

- **Boyce-Codd Normal Form (BCNF)**: BCNF is a stricter form of normalization compared to 3NF. A relation is in BCNF if, for every non-trivial functional dependency X → Y, X is a superkey. In other words, every determinant (X) must be a candidate key. BCNF eliminates certain types of anomalies that may still exist in 3NF, particularly those related to overlapping candidate keys.

41. **JOIN dependencies**: Join dependencies are constraints that involve multiple relations (tables) in a database. They specify relationships between tuples from different relations, typically based on common attributes or keys. Join dependencies are relevant in the context of decomposition of relations during normalization.

42. **Inclusion dependencies**: Inclusion dependencies specify relationships between columns or sets of columns within a single relation (table). They indicate that the values appearing in one set of columns must also appear in another set of columns.

43. **Template dependencies**: Template dependencies are constraints that specify patterns or templates for the values of attributes in a relation. They define rules about the permissible combinations of values in certain attributes based on the values in other attributes.

44. **Multivalued dependencies**: Multivalued dependencies occur when there is a dependency between sets of attributes in a relation such that the presence of certain values implies the presence of other values, but without implying functional dependency. They are typically addressed during normalization to eliminate redundancy and improve data integrity.

45. **Project Join Normal form**: Project-Join Normal Form (PJNF) is a form of normalization that ensures the elimination of redundancy and the preservation of join dependencies. A relation is in PJNF if it is in BCNF and all its join dependencies are implied by the candidate keys. PJNF addresses the need to preserve certain join dependencies that may be lost during normalization into BCNF.

46. **Domain Key Normal Form (DKNF)**:

Domain Key Normal Form (DKNF) is a level of normalization in which all constraints are expressed purely as domain constraints and key constraints. In DKNF, every constraint must be a logical consequence of the definition of the keys and domains. It ensures that each constraint in the database is a natural consequence of the data model and not an artifact of how the data is stored or represented.

47. **Informal guidelines for database design**:

- Identify the purpose and scope of the database.
- Analyze the data requirements and relationships.
- Normalize the database to reduce redundancy.
- Choose appropriate data types and constraints.
- Design clear and consistent naming conventions.
- Ensure data integrity through constraints and validation rules.
- Optimize performance through proper indexing and tuning.
- Consider security and access control requirements.
- Document the database schema and design decisions for future reference.

48. **Super key**:

A super key is a set of one or more attributes (columns) that uniquely identifies each tuple (row) in a relation (table). It is a superset of candidate keys and may contain extra attributes. Super keys help in identifying unique rows in a relation.

49. **Primary key**:

A primary key is a special type of super key that uniquely identifies each tuple in a relation and ensures entity integrity. It cannot contain NULL values and must be unique for each tuple. Only one primary key is allowed per relation.

50. **Foreign key**:

A foreign key is an attribute or set of attributes in one relation that refers to the primary key in another relation. It establishes a relationship between two tables by enforcing referential integrity, ensuring that the values in the foreign key column(s) match the values in the primary key column(s) of the referenced table.

51. **Unique key**:

A unique key is a constraint that ensures that all values in a column or combination of columns are unique, similar to a primary key. However, unlike a primary key, a table can have multiple unique keys, and they can contain NULL values.

52. **Prime attribute**:

A prime attribute is an attribute that is part of a candidate key, i.e., it is a member of at least one candidate key for a relation. Prime attributes contribute to determining the uniqueness of tuples in a relation.

53. **Trivial functional dependency**:

A trivial functional dependency occurs when the dependent attribute(s) are determined by the same set of attributes that they are part of. In other words, the dependency holds trivially because it is already implied by the definition of the attributes.

54. **Fully functional dependency**:

A functional dependency is said to be fully functional if removing any attribute from the determinant results in a violation of the dependency. In other words, every attribute in the determinant is necessary for the dependency to hold.

55. **Armstrong’s inference rules**:

- Reflexivity

- Augmentation

- Transitivity

56. **Soundness and completeness of Armstrong’s inference rules**:

Armstrong's inference rules are considered sound because they always yield true functional dependencies based on the given set of dependencies. They are also complete because they can derive all possible functional dependencies that logically follow from the given set.

57. **Denormalization**:

Denormalization is the process of intentionally introducing redundancy into a database design for the purpose of improving query performance or simplifying data retrieval. It involves relaxing normalization principles to optimize certain aspects of database performance.

58. **Transaction**:

A transaction is a logical unit of work that represents a series of database operations (such as reads, writes, or updates) that must be executed as a single indivisible unit. Transactions ensure data consistency and integrity by either completing all operations successfully (commit) or rolling back to the previous state (rollback) in case of failure.

59. **ACID properties**:

- Atomicity: Ensures that a transaction is treated as a single unit of work, either all its operations are executed successfully or none.

- Consistency: Ensures that the database remains in a consistent state before and after the transaction.

- Isolation: Ensures that concurrent transactions do not interfere with each other's execution or affect their outcomes.

- Durability: Ensures that the changes made by a committed transaction are permanently saved and survive system failures.

60. **Schedule**:

A schedule in database management represents the order in which transactions are executed in a multi-user environment. It specifies the sequence of operations performed by various transactions and their respective timings.

61. **DBMS vs. file-based systems**:

While DBMS offers numerous advantages, there are situations where file-based systems may be more suitable, such as:

- Small-scale applications with limited data and simple data access requirements.

- Applications where performance overhead of a DBMS is a concern.

- Situations where data structures are simple and static.

- When the cost of implementing and maintaining a DBMS outweighs its benefits.

62. **3-level architecture of DBMS**:

- **External Level (View Level)**: This is the highest level of abstraction where users interact with the database. Users define their views of the database using languages like SQL.

- **Conceptual Level (Logical Level)**: This level represents the logical structure of the entire database, including its schema, relationships, and constraints. It hides the physical implementation details from users and provides data independence.

- **Internal Level (Physical Level)**: This level deals with the physical storage and organization of data on the storage devices. It includes data structures, indexing methods, and access paths optimized for efficient data retrieval and storage.

63. **Logical vs. Physical architecture of DBMS**:

- **Logical architecture**: Defines the structure and organization of data in the database from a logical perspective, including schemas, relationships, and constraints.

- **Physical architecture**: Concerns with the actual storage and retrieval of data on the physical storage devices, including data files, indexes, and access methods. It deals with optimization techniques for efficient data storage and retrieval.

64. *(Please provide more details or specify any specific requirements for creating an ER diagram and relational schema.)*

65. **Utility of relational algebra & relational calculus**:

Relational algebra and relational calculus are theoretical foundations of relational databases used for query formulation and optimization. They provide a formal framework for expressing relational operations and queries. Some software based on these concepts include relational database management systems like PostgreSQL, MySQL, Oracle, and SQL Server.

66. **Set theory's contribution to RDBMS**:

Set theory forms the basis of relational database theory, providing concepts such as relations, sets, tuples, and operations like union, intersection, and difference. RDBMS uses set theory to model and manipulate data, ensuring data integrity and consistency. For example, the UNION operation in relational algebra corresponds to the union operation in set theory, which combines the elements of two sets without duplicates.

67. **Redundancy of data**:

Redundancy in data can be beneficial in certain situations, such as improving query performance by precomputing and storing derived data or enhancing fault tolerance by replicating critical information across multiple locations. However, excessive redundancy can lead to inconsistency, wasted storage space, and maintenance complexities. To mitigate these issues, it's essential to carefully manage redundancy and ensure data integrity through normalization and proper design.

68. In Oracle, the version numbers such as Oracle 8, Oracle 9, etc., indicate the major release versions of the Oracle Database. Each new version typically introduces significant enhancements, features, and improvements over the previous version. The "i" suffix in versions like Oracle 8i, Oracle 9i, etc., stands for "Internet" and was introduced to signify the internet-related enhancements and features added to the respective versions.

69. **File organization techniques**:

- **Sequential file organization**: Records are stored in sequence according to a primary key field. Suitable for applications requiring sequential access.

- **Indexed file organization**: Records are stored in a sequential order, and an index is created to facilitate direct access based on key values.

- **Hashed file organization**: Records are stored in a hash table structure, and a hash function is used to determine the storage location based on the key value. Suitable for applications requiring quick access to records based on equality searches.

- **Clustered file organization**: Similar records are physically stored together on disk to improve retrieval efficiency.

- **Heap file organization**: Records are stored in no particular order, and pointers or offsets are used to navigate through the file.

Difference between Binary Tree, B-tree, and B+ tree:

- **Binary Tree**: A binary tree is a tree data structure where each node has at most two children. It is typically used in memory structures and is not directly used as a file organization technique in databases.

- **B-tree**: A B-tree is a self-balancing tree data structure that maintains sorted data and allows searches, sequential access, insertions, and deletions in logarithmic time. It is commonly used in database systems for indexing.

- **B+ tree**: A B+ tree is a variant of the B-tree where all the data is stored in the leaf nodes, and internal nodes only contain keys for navigation. B+ trees are optimized for disk storage systems and are widely used in database indexing due to their ability to support range queries, efficient sequential access, and high fanout.

When to use B-tree or B+ tree:

- Use B-tree when the data is primarily stored in memory or when the dataset is small enough to fit in memory.

- Use B+ tree when dealing with large datasets that need to be stored on disk, as B+ trees have better disk I/O performance due to their optimized structure.

70. **Functional dependencies to remove for achieving respective normal forms**:

- **First Normal Form (1NF)**: No functional dependencies need to be removed.

- **Second Normal Form (2NF)**: Remove partial dependencies.

- **Third Normal Form (3NF)**: Remove transitive dependencies.

- **Boyce-Codd Normal Form (BCNF)**: Remove non-trivial dependencies where the determinant is not a candidate key.

- **Fourth Normal Form (4NF)**: Remove multi-valued dependencies.

71. **Mathematical basis of SQL**:

SQL (Structured Query Language) is based on relational algebra and relational calculus, which are mathematical formalisms for dealing with relational databases. SQL statements such as SELECT, PROJECT, JOIN, etc., correspond to operations in relational algebra. The SQL statement `SELECT * FROM student` performs like a projection operation, selecting all columns (attributes) from the "student" relation (table), similar to the projection operation in relational algebra, which selects certain columns from a relation.