

Questions and Answers

SHORT ANSWER QUESTIONS:

Q1: What is Data?

A1: Data is a collection of raw, unorganized facts and details that do not carry any specific purpose or significance until processed.

Q2: What is the difference between Data and Information?

A2: Data is raw and unorganized, while information is processed, organized, and structured data that provides context and enables decision making.

Q3: What is a Database?

A3: A database is an electronic system where data is stored in a way that allows for easy access, management, and updates.

Q4: What is DBMS?

A4: A Database Management System (DBMS) is a collection of interrelated data and a set of programs to access and manage that data efficiently.

LONG ANSWER QUESTIONS:

Q1: Explain the Three Schema Architecture in DBMS.

A1: The Three Schema Architecture in DBMS consists of the Physical, Logical, and View levels. The Physical level describes how data is stored, including storage allocation and structure. The Logical level focuses on the design of the database and relationships between data. The View level provides a personalized view for different user groups, hiding unnecessary details. This architecture allows for data abstraction and efficient data access for multiple users.

Q2: Discuss the concept of Specialisation and Generalisation in Extended ER Features.

A2: Specialisation in Extended ER Features involves subgrouping an entity set into distinct sub entity sets based on specific functionalities or features. It is a top-down approach that helps refine the database schema by showing distinctive attributes of sub entities. Generalisation, on the other hand, is the reverse process where sub entity sets are combined back into a parent entity set. These concepts are used to model complex database structures and relationships effectively.

SHORT ANSWER QUESTIONS:

Q1: What is the purpose of Specialisation in a database design?

A1: Specialisation is used to group entities with unique characteristics and refine the overall database blueprint.

Q2: How is Generalisation different from Specialisation in a database design?

A2: Generalisation is the reverse process of Specialisation, where common attributes of entities are grouped into a superclass to avoid data redundancy.

Q3: What is Attribute Inheritance in database design?

A3: Attribute Inheritance is the concept where lower-level entity sets inherit attributes from higher-level entity sets in both Specialisation and Generalisation.

Q4: How does Participation Inheritance work in database design?

A4: Participation Inheritance states that if a parent entity set participates in a relationship, its child entity sets will also participate in that relationship.

Q5: What is the purpose of Aggregation in database design?

A5: Aggregation is used to show relationships among relationships by treating them as higher-level entities, helping to avoid redundancy in the database.

LONG ANSWER QUESTIONS:

Q1: Explain the relationship between Specialisation and Generalisation in a database design and provide an example.

A1: Specialisation and Generalisation are techniques used to refine a database schema by grouping entities with unique characteristics or common attributes. For example, in a transportation database, entities like Car, Jeep, and Bus can be specialised into a superclass "Vehicle" to avoid repeating common attributes like number of wheels or engine type.

Q2: Describe the concept of Relational Keys in a Relational Model and explain the types of keys used.

A2: Relational Keys are used to uniquely identify tuples in a relation. The types of keys include Super Key (SK), Candidate Key (CK), Primary Key (PK), Alternate Key (AK), Foreign Key (FK), Composite Key, Compound Key, and Surrogate Key. Each key serves a specific purpose in maintaining data integrity and relationships between

tables.

Q3: Discuss the importance of SQL in database management and explain the different types of SQL commands.

A3: SQL (Structured Query Language) is essential for accessing and manipulating data in a relational database. It includes commands like DDL (Data Definition Language) for defining relation schema, DRL/DQL (Data Retrieval Language / Data Query Language) for retrieving data, DML (Data Modification Language) for modifying data, DCL (Data Control Language) for granting or revoking authorities, and TCL (Transaction Control Language) for managing transactions. Each type of command plays a crucial role in managing and querying databases effectively.

SHORT ANSWER QUESTIONS:

Q1: What does the SQL keyword "BETWEEN" do?

A1: The "BETWEEN" keyword in SQL is used to select rows based on a range of values. For example, "SELECT * FROM customer WHERE age BETWEEN 0 AND 100;" will retrieve rows where the age falls within the range of 0 to 100, inclusive.

Q2: Explain the use of the SQL keyword "GROUP BY."

A2: The "GROUP BY" clause in SQL is used to group data from multiple records based on one or more columns. It is typically used with aggregation functions like COUNT(), SUM(), AVG(), MIN(), and MAX() to perform actions on grouped data.

Q3: What is the purpose of SQL views?

A3: SQL views are virtual tables created by queries that join one or more tables. They do not contain data of their own but provide a way to simplify complex queries, improve security, and ensure data consistency by reflecting changes in the underlying tables.

LONG ANSWER QUESTIONS:

Q1: Explain the concept of database normalization and its importance in database design.

A1: Database normalization is a process used to organize a database structure efficiently by reducing redundancy and dependency of data. It involves breaking down tables into smaller, related tables to minimize anomalies like insertion, update, and deletion issues. Normalization helps improve data integrity, reduce storage space, and enhance database performance.

Q2: Compare and contrast the different types of SQL joins, including INNER JOIN, OUTER JOIN, CROSS JOIN, and SELF JOIN.

A2: SQL joins are used to combine data from multiple tables based on a specific condition. INNER JOIN returns rows with matching values in both tables, OUTER JOIN includes unmatched rows from one or both tables, CROSS JOIN produces the Cartesian product of the tables, and SELF JOIN joins a table to itself. Each type of join has its own use cases and implications on the resulting dataset.

SHORT ANSWER QUESTIONS:

Q1: What is the primary focus of 1NF in database normalization?

A1: Ensuring that every relation cell has an atomic value and that the relation does not have multi-valued attributes.

Q2: What is the key requirement for a relation to be in 2NF?

A2: The relation must be in 1NF and should not have any partial dependencies, with all non-prime attributes being fully dependent on the primary key.

Q3: Explain the concept of Atomicity in the context of transactions.

A3: Atomicity ensures that either all operations of a transaction are reflected properly in the database, or none are, to maintain data integrity.

Q4: How does the shadow-copy scheme contribute to ensuring atomicity and durability in transactions?

A4: The shadow-copy scheme involves creating copies of the database and updating them separately, allowing for easy rollback in case of transaction failures, thus ensuring atomicity and durability.

LONG ANSWER QUESTIONS:

Q1: Describe the characteristics and advantages of NoSQL databases over traditional relational databases.

A1: NoSQL databases are schema-free, handle unstructured data efficiently, scale easily, and offer flexible data models. They excel in storing large amounts of data and provide high availability, horizontal scaling, and faster read operations. These databases are well-suited for agile development, cloud applications, and scenarios requiring scale-out architecture.

Q2: Explain the concept of Transaction in databases, including its states and the importance of ACID properties.

A2: A transaction is a logical unit of work in a database that consists of one or more SQL statements. It can be in different states like active, partially committed,

committed, failed, aborted, or terminated. ACID properties - Atomicity, Consistency, Isolation, and Durability - ensure data integrity by guaranteeing that transactions are completed successfully or rolled back in case of failures, maintaining the consistency and reliability of the database system.

SHORT ANSWER QUESTIONS:

Q1: What is the primary advantage of NoSQL databases over traditional relational databases?

A1: NoSQL databases provide flexible schemas and scale easily with large amounts of data and high user loads.

Q2: Name one type of NoSQL data model.

A2: Key-Value Stores.

Q3: What is the key difference between SQL and NoSQL databases in terms of data storage model?

A3: SQL databases store data in tables with fixed rows and columns, while NoSQL databases store data in various formats like documents, key-value pairs, wide-columns, or graphs.

LONG ANSWER QUESTIONS:

Q1: Explain the concept of key-value stores in NoSQL databases and provide examples of key-value databases.

A1: Key-value stores are the simplest type of NoSQL databases where data is stored as key-value pairs. These databases are efficient for systems that need to quickly retrieve data. Examples of key-value databases include Oracle NoSQL, Amazon DynamoDB, MongoDB, and Redis.

Q2: Compare and contrast relational databases with NoSQL databases in terms of data storage model, development history, and primary purpose.

A2: Relational databases store data in tables with fixed rows and columns, were developed in the 1970s to reduce data duplication, and are general-purpose databases primarily using SQL. On the other hand, NoSQL databases store data in various formats, were developed in the late 2000s to scale easily and support rapid application changes, and have different types like document, key-value, wide-column, and graph databases with specific purposes like large data with simple queries, analytics, or relationship analysis.

SHORT ANSWER QUESTIONS:

Q1: What is the purpose of load balancing in database clustering?

A1: Load balancing ensures that traffic is evenly distributed among multiple machines to prevent overload on any one machine, thus maintaining high availability and preventing slowdowns.

Q2: How does partitioning help in managing large database tables?

A2: Partitioning divides a large database into smaller, manageable slices called partitions, allowing for more efficient data handling without the need to process the entire database at once.

Q3: What is the main idea behind sharding in database optimization?

A3: Sharding involves splitting up data across multiple database instances and introducing a routing layer to direct requests to the appropriate instances, improving scalability and availability.

Q4: What is the difference between vertical and horizontal partitioning in database optimization?

A4: Vertical partitioning slices data columns-wise, while horizontal partitioning slices data row-wise. Vertical partitioning requires accessing different servers for complete tuples, whereas horizontal partitioning stores independent chunks of data tuples in different servers.

LONG ANSWER QUESTIONS:

Q1: Explain the concept of high availability in database clustering and how it is achieved through load balancing and extra machines.

A1: High availability refers to the amount of time a database is considered available. Database clustering can achieve high levels of availability by evenly distributing traffic among multiple machines using load balancing. With extra machines in place, if one server fails, the database remains accessible as requests are redirected to other functioning nodes, ensuring continuous availability.

Q2: Discuss the advantages of partitioning in database optimization and when it is typically applied.

A2: Partitioning offers advantages such as parallelism, availability, performance, manageability, and cost reduction. It is applied when datasets become too large to manage efficiently, when the number of requests overwhelms a single database server, or when scaling up vertically becomes costly. By dividing data into smaller partitions, partitioning allows for better performance, easier management, and cost-effective scaling of databases.

SHORT ANSWER QUESTIONS:

Q1: What is the CAP Theorem?

A1: The CAP Theorem states that a distributed system can only provide two of three properties simultaneously: consistency, availability, and partition tolerance.

Q2: What is the purpose of Command Query Responsibility Segregation (CQRS)?

A2: CQRS separates read and write operations physically, allowing different machines to handle specific tasks to improve system performance and scalability.

Q3: What is the Master-Slave database concept?

A3: The Master-Slave concept involves directing write operations to a master database while reading operations are carried out from slave databases to optimize system performance and reliability.

LONG ANSWER QUESTIONS:

Q1: Explain the different patterns for scaling a database system mentioned in the text.

A1: The text outlines several patterns for scaling a database system. Query optimization and connection pool implementation involve caching data and using connection pools to improve performance. Vertical scaling involves upgrading hardware to handle increased load. CQRS separates read and write operations to better manage system resources. Multi-primary replication distributes write requests across multiple nodes. Partitioning data by functionality involves separating data into different databases for better organization. Horizontal scaling (sharding) distributes data across multiple machines. Data center wise partitioning helps distribute traffic across multiple data centers for improved performance.

Q2: Discuss the tradeoffs between consistency, availability, and partition tolerance in the context of the CAP Theorem.

A2: The CAP Theorem states that in a distributed system, you can only achieve two out of three properties: consistency, availability, and partition tolerance. Consistency ensures that all nodes see the same data simultaneously, availability guarantees that the system remains operational all the time, and partition tolerance allows the system to continue functioning even if there are communication breakdowns between nodes. Depending on the system's requirements, designers must make tradeoffs between these properties to ensure optimal performance and reliability.