Questions and Answers

SHORT ANSWER QUESTIONS:

Q1: What is the difference between data and information?

A1: Data is a collection of raw, unorganized facts and details, while information is processed, organized, and structured data that provides context and enables decision making.

Q2: What is a DBMS?

A2: A DBMS, or Database Management System, is a collection of interrelated data and a set of programs to access and manage that data efficiently.

Q3: What are the types of data mentioned in the text?

A3: The types of data mentioned in the text are quantitative data (numerical form like weight, volume, cost) and qualitative data (descriptive, non-numerical information like name, gender).

LONG ANSWER QUESTIONS:

Q1: Explain the concept of Specialisation in the Extended ER Features.

A1: Specialisation in the Extended ER Features refers to the process of subgrouping an entity set into further sub-entity sets based on their distinct functionalities, specialities, and features. It is a top-down approach where the parent entity set is divided into more specific sub-entity sets known as subclasses. This division allows for a more detailed representation of the entities within the database schema. For example, a Person entity set can be specialised into Customer, Student, and Employee subclasses. Specialisation is represented by a triangle component in the ER diagram and establishes an "is-a" relationship between the superclass and its subclasses. The main purpose of specialisation is to highlight the unique characteristics and attributes of the sub-entities, making the database design more refined and tailored to specific requirements.

Q2: Discuss the Three Schema Architecture in DBMS.

A2: The Three Schema Architecture in DBMS is designed to provide users with an abstract view of the data while hiding the details of how the data is stored and maintained. It consists of three levels: the Physical level/Internal level, the Logical level/Conceptual level, and the View level/External level.

- The Physical level describes how data is stored, including low-level data structures and physical schema details like storage allocation and data compression.

- The Logical level defines the design of the database at a conceptual level, specifying what data is stored and the relationships between the data. It is used by database administrators to make decisions about the information to be stored.

- The View level provides different views of the database for different user groups, simplifying user interaction with the system. Each view schema describes a specific part of the database that a particular user group is interested in, hiding the rest of the database. Subschemas are used to describe these different views and provide a security mechanism to control access to specific parts of the database. This architecture enables multiple users to access the same data with personalized views while storing the underlying data only once.

SHORT ANSWER QUESTIONS:

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

A1: Specialisation is used to group entities with specific attributes together in order to refine the database blueprint.

Q2: What is the difference between a Super Key and a Candidate Key in a relational database?

A2: A Super Key is any set of attributes that uniquely identifies each tuple, while a Candidate Key is the minimum subset of super keys that can uniquely identify each tuple without any redundant attributes.

Q3: What is the purpose of Referential Constraints in a database?

A3: Referential Constraints help maintain consistency among tuples of two relations by ensuring that values in specified attributes of a referencing relation also appear in the specified attributes of the referenced relation.

LONG ANSWER QUESTIONS:

Q1: Explain the process of converting an ER diagram to a Relational Model.

A1: When converting an ER diagram to a Relational Model, strong entities are represented as individual tables with the entity name, attributes become columns, and the entity's Primary Key (PK) is used as the relation's PK. Weak entities are formed into tables with all attributes and the PK of the corresponding strong entity as a FK. Single value attributes are represented as columns, while composite attributes are split into separate attributes in the relation. Multivalued attributes result in new tables with the entity's PK as a FK and a column for each value. Derived attributes are not considered. Generalisation can be represented by creating tables for higher level

entity sets and lower level entity sets, or by creating separate tables for each lower level entity set with all attributes of that entity set plus attributes of the higher-level entity set.

Q2: Discuss the key constraints present in a database management system.

A2: The six types of key constraints in a DBMS are NOT NULL, UNIQUE, DEFAULT, CHECK, PRIMARY KEY, and FOREIGN KEY. NOT NULL ensures no NULL values, UNIQUE ensures all values in a column are different, DEFAULT sets default values, CHECK maintains data integrity, PRIMARY KEY uniquely identifies each entity, and FOREIGN KEY establishes relationships between tables. Each key constraint plays a crucial role in maintaining data consistency and integrity within the database.

SHORT ANSWER QUESTIONS:

Q1: What is the purpose of the GROUP BY clause in SQL?

A1: The GROUP BY clause in SQL is used to collect data from multiple records and group the result by one or more columns. It is generally used in a SELECT statement to group data into categories based on the column given.

Q2: What does the IS NULL operator do in SQL?

A2: The IS NULL operator in SQL is used to filter records where a specific column contains a NULL value.

Q3: What is the difference between HAVING and WHERE clauses in SQL?

A3: The WHERE clause is used to filter rows from a table based on specified conditions, while the HAVING clause is used to filter rows from groups based on specified conditions. HAVING is used after the GROUP BY clause, while WHERE is used before the GROUP BY clause.

LONG ANSWER QUESTIONS:

Q1: Explain the concept of normalization in databases and why it is important.

A1: Normalization is a process used to minimize data redundancy and avoid anomalies in a database. It involves dividing composite attributes into individual attributes and linking tables using relationships. Normalization helps in eliminating undesirable characteristics like Insertion, Update, and Deletion Anomalies. By organizing data into different normal forms, redundancy is reduced, and data integrity is maintained. This ensures that the database is efficient, easy to maintain, and free from anomalies that can affect data accuracy.

Q2: Describe the difference between INNER JOIN and OUTER JOIN in SQL.

A2: INNER JOIN in SQL returns a resultant table that has matching values from both tables, while OUTER JOIN returns a resulting table that includes all data from one table and only matched data from the other table. Within OUTER JOIN, there are three types: LEFT JOIN returns all data from the left table and matched data from the right table, RIGHT JOIN returns all data from the right table and matched data from the left table, and FULL JOIN returns all data when there is a match on either the left or right table. These JOIN types are important for combining data from multiple tables in a relational database system.

SHORT ANSWER QUESTIONS:

Q1: What is the primary purpose of normalisation in a database?

A1: The primary purpose of normalisation in a database is to minimise data redundancy and ensure data consistency.

Q2: What are the ACID properties of a transaction?

A2: The ACID properties of a transaction are Atomicity, Consistency, Isolation, and Durability.

Q3: What is the main advantage of using NoSQL databases?

A3: The main advantage of using NoSQL databases is their ability to handle huge amounts of unstructured data and scale easily.

LONG ANSWER QUESTIONS:

Q1: Explain the concept of normalisation in a database and discuss the different types of normal forms mentioned in the text.

A1: Normalisation in a database is the process of organizing data in a way that reduces redundancy and dependency. The different types of normal forms include:

1. 1NF: Ensures that every relation cell has atomic values and does not have multi-valued attributes.

2. 2NF: Requires that the relation be in 1NF and eliminates partial dependencies by ensuring all non-prime attributes are fully dependent on the primary key.

3. 3NF: Builds on 2NF by removing transitive dependencies, ensuring that non-prime attributes do not determine other non-prime attributes.

4. BCNF: Requires the relation to be in 3NF and enforces that functional dependencies are such that A must be a super key.

Q2: Describe the ACID properties of a transaction and explain how they ensure data integrity.

A2: The ACID properties of a transaction are:

1. Atomicity: Ensures that either all operations of a transaction are reflected properly in the database, or none are.

2. Consistency: Requires that integrity constraints are maintained before and after a transaction, ensuring the database remains consistent.

3. Isolation: Guarantees that each transaction is unaware of other transactions executing concurrently, maintaining data integrity.

4. Durability: Ensures that once a transaction completes successfully, the changes made to the database persist even in the event of system failures. These properties collectively ensure data integrity and consistency in a database system.

SHORT ANSWER QUESTIONS:

Q1: What is a key-value store in NoSQL databases?

A1: A key-value store is the simplest type of NoSQL database where data elements are stored as key-value pairs, consisting of an attribute name (key) and a value. Examples include Oracle NoSQL, Amazon DynamoDB, MongoDB, and Redis.

Q2: What is a disadvantage of NoSQL databases according to the text?

A2: According to the text, a disadvantage of NoSQL databases is that data redundancy can occur since data models are optimized for queries and not for reducing data duplication. This can lead to larger storage requirements compared to SQL databases.

LONG ANSWER QUESTIONS:

Q1: Explain the differences between SQL and NoSQL databases as discussed in the text.

A1: SQL databases store data in tables with fixed rows and columns, have a long history dating back to the 1970s, and support ACID properties. Examples include Oracle, MySQL, and PostgreSQL. NoSQL databases, on the other hand, store data differently and come in various types like document, key-value, wide-column, and graph. They are schema-free, flexible, and can handle large amounts of data and high user loads. Examples include MongoDB, Redis, Cassandra, and Neo4j. NoSQL databases were developed in the late 2000s with a focus on scaling and rapid application changes driven by agile practices. They do not support ACID properties

universally and typically do not require joins for queries.

Q2: What are the advantages of using a key-value store in a NoSQL database?

A2: Key-value stores in NoSQL databases offer advantages such as fast and easy access to data through key-based queries, making them ideal for real-time random data access like user session attributes in online applications such as gaming or finance. They also serve as efficient caching mechanisms for frequently accessed data or configuration based on keys. Additionally, key-value databases use compact and efficient index structures to quickly and reliably locate a value by its key, allowing for constant time data retrieval.

SHORT ANSWER QUESTIONS:

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

A1: Load balancing ensures that traffic is evenly distributed across multiple machines, preventing any single machine from becoming overworked and slowing down traffic.

Q2: What is sharding?

A2: Sharding is a technique used to implement horizontal partitioning in databases, where data is split across multiple database instances to improve scalability and performance.

LONG ANSWER QUESTIONS:

Q1: Explain the concept of high availability in database clustering and how it is achieved.

A1: High availability refers to the amount of time a database is considered available for use. In database clustering, high availability is achieved through load balancing and having extra machines ready to take over in case one fails. By spreading the workload across multiple machines, the system can continue to function even if one machine goes down. This ensures that the database remains accessible to users and prevents any downtime or loss of data. The level of availability needed depends on the number of transactions and analytics being performed on the data.

Q2: Describe the advantages of partitioning in database management and when it should be applied.

A2: Partitioning is the technique of dividing a large database into smaller, more manageable slices of data called partitions. This helps in improving performance, scalability, and manageability of the database. When datasets become too large to handle efficiently or when the system's response time is affected by the volume of requests, partitioning should be applied. By dividing the data into smaller chunks, parallelism is increased, availability is enhanced, performance is optimized, and costs

are reduced. Additionally, partitioning allows for easier management of large database tables and improves the overall efficiency of the database system.

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 the Master-Slave database concept?

A2: The purpose of the Master-Slave database concept is to optimize IO in a system where the number of requests exceeds what a single DB server can handle efficiently.

LONG ANSWER QUESTIONS:

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

A1: The text outlines several patterns for scaling a database system:

- Pattern 1 involves query optimization and connection pool implementation, which includes caching frequently used data and introducing database redundancy.

- Pattern 2 focuses on vertical scaling or scale-up, which involves upgrading the existing hardware to handle increased load.

- Pattern 3 discusses Command Query Responsibility Segregation (CQRS), where read and write operations are separated physically to handle increased load.

- Pattern 4 involves multi-primary replication, distributing write requests to replicas in a logical circular ring.

- Pattern 5 suggests partitioning data by functionality, separating data into different schemas or machines.

- Pattern 6 discusses horizontal scaling or scale-out, using sharding to allocate data across multiple machines.

- Pattern 7 involves data center wise partitioning, distributing traffic across data centers for better availability and disaster recovery.

Q2: How does the CAP theorem impact the choice of database systems?

A2: The CAP theorem states that a distributed system can only provide two of three properties simultaneously: consistency, availability, and partition tolerance. This impacts the choice of database systems as it forces developers to prioritize which properties are most important for their application. For example, CA databases prioritize consistency and availability but may not be practical for all scenarios due to the inevitability of partitions. CP databases prioritize consistency and partition tolerance, while AP databases prioritize availability and partition tolerance. Understanding the CAP theorem helps in selecting the right database system based on the specific needs of the application.