**Database Design Assignment**

1. **Physical Entity Relationship diagram of database**

A screenshot of a computer

Description automatically generated

**2. Explain about searching performance. How will you handle replication in SQL for searching & Reporting?**

Finding and obtaining specific data from a database based on predetermined criteria or conditions is referred to as searching. For big datasets to be swiftly searched and retrieved with relevant data, efficient searching is necessary. Optimising search operations involves a number of methods and elements:

Indexing: The process of establishing data structures (or "indices") that the database management system (DBMS) can use to identify data quickly is known as indexing. On columns that are regularly utilised in search queries, indexes are often generated. Bitmap, hash, and B-tree indexes are examples of common index types.

Replications in Database Design: The process of making and keeping backup copies of a database or particular database tables across various servers or locations is known as replication. Replication is primarily used to increase performance, fault tolerance, and data availability.

Depending on whether changes are instantaneously reflected in all replicas (synchronous) or whether there is a minor propagation delay (asynchronous), replication can be either synchronous or asynchronous. Each replication method includes trade-offs with regard to complexity, performance, and data consistency.

Replication and searching are crucial components of database design, to sum up. Effective data retrieval is ensured through efficient searching, while replication improves data availability and fault tolerance by keeping redundant copies of the database. For the purpose of developing scalable, dependable, and high-performance database systems, these notions must be implemented properly.

In order to improve search performance and speed up data retrieval, this database design places a strong emphasis on data replication and indexing.

**3. Explain what major factors are taken into consideration for performance.**

To ensure effective data storage, retrieval, and processing, it is necessary to take a number of aspects into account while designing a high-performance database. The following are some important considerations for this database design:

• Designing a data model and schema The relationships between the entities are appropriately reflected in this database schema. Denormalize the data strategically to improve query efficiency while normalising the data to lessen duplication and anomalies.

• Indexing: Recognised and made appropriate indexes for the columns frequently used in search criteria. Indexes enable the database to easily find pertinent data, which speeds up data retrieval. Finding the appropriate balance is essential because having too many indexes might negatively affect insert/update performance.

• Data Types and Storage: To maximise processing and storage, the right data types were selected for each column. Use date/time data types for information about dates, and integer data types for numerical data, as an example. Be aware of your storage limitations and, if appropriate, think about data compression.

• Partitioning: Based on predetermined criteria, divided huge tables into more manageable, smaller segments. Because partitioning enables the DBMS to access and handle smaller portions of data, it can improve data retrieval and maintenance processes.

• Normalisation and Denormalization: Normalisation was used to improve data integrity and minimise redundancy in the data. However, when necessary, especially in read-heavy systems, denormalize some tables for better query speed.

• Replication and Load Balancing: Read and write operations were split over a number of servers using replication and load balancing techniques. Data replication enhances it

**4. Mention about Indexing, Normalization and Denormalization.**

Ans.

Indexing: The most commonly used columns in search criteria were identified, and the relevant indexes were built. Indexes enable the database to easily find pertinent data, which speeds up data retrieval. Finding the appropriate balance is essential because having too many indexes might negatively affect insert/update performance.

Normalisation:

Data organisation in a database is done by normalisation. It entails the creation of tables and the establishment of linkages between those tables in accordance with rules intended to safeguard the data and increase the database's adaptability by removing duplication and inconsistent reliance.

The basic goals of normalisation are to reduce data duplication and increase data consistency, which improve data administration and querying effectiveness.

Larger tables in this database are divided into smaller ones as part of the normalisation process.

Denormalization is the deliberate introduction of redundancy into a relational database by the consolidation of data from several related tables into one table or the duplication of data across several tables. Denormalization aims to increase query performance and data retrieval effectiveness, particularly for workloads that involve a lot of reading. When normalised structures alone may not be enough to meet the performance needs of the application, this technique is utilised to optimise the database design.

After a database has been normalised to a higher normal form, denormalization is typically used. Normalisation can result in complex joins and poorer query performance, especially in situations where frequent and complex joins are needed to get data from numerous tables, even while it lowers data redundancy and upholds data integrity.

**5. How will you handle scaling, if required at any point of time**

In order to support growing data quantities and user demands as your application scales, handling scalability in database design is crucial. Depending on your particular needs and the type of database you're using, scaling can be accomplished via a variety of ways. Here are a few typical methods for dealing with scaling in database design:

• Vertical scaling, also known as scaling up, entails upgrading the hardware resources of your current database server. Typically, this entails boosting the server's CPU performance, memory, or storage capacity. Although vertical scaling is frequently simple, there are times when it cannot be done because of the capacity of a single server. It works better for applications with moderate scaling requirements.

• Horizontal scaling (also known as scaling out): Horizontal scaling entails the addition of more servers in order to disperse the workload and data over multiple machines. By utilising this strategy, you may handle rising traffic and data quantities by adding extra servers as necessary. In general, horizontal scaling is more adaptable and scalable than vertical scaling. For managing complex systems and applications with significant development rates, it is the preferred method.

1. Replication: Replication is the process of making numerous copies (also known as replicas) of the database on other servers. Replication improves data availability and fault tolerance and makes load balancing for applications that require a lot of reading possible. Common replication configurations include master-slave replication and master-master replication.

• Load Balancing: To uniformly divide incoming queries among several database servers, use load balancing technologies. Load balancers make ensuring that the burden is distributed among multiple servers to prevent overloading.

• Auto-scaling: Use cloud computing platforms to benefit from features for auto-scaling. These platforms provide the ability to automatically add or remove database instances in response to demand, assuring efficient resource use and low costs.

We can scale this application using the database design, making numerous copies of the database across other servers, which will Replication improves data availability and fault tolerance and makes load balancing for applications that require a lot of reading possible. In order to accommodate the users' growing need, horizontal and vertical scaling can also be used at any time in this database.

**6. Mention all the assumptions you are taking for solutions.**

To assure the efficacy, efficiency, and stability of the database system, database designers and developers frequently make a number of assumptions when creating a database. Depending on the particular project requirements and the kind of database being developed, these assumptions may change. The following are some typical database design presumptions:

• Data Integrity: The accuracy, consistency, and validity of the data recorded in the database are presumptions. To prohibit the addition of inaccurate or invalid data, this database will enforce data integrity requirements such as primary keys, foreign keys, unique constraints, and check constraints.

• Consistency: The database design makes the assumption that the data stored in the database will uphold consistency, which entails that it will abide by established guidelines and limitations.

• Concurrency Control: Multiple users may access and edit data concurrently, which is a presumption made in the database design. It should therefore include the proper concurrency management methods to stop data inconsistencies brought on by concurrent access.

• Backup and recovery: Regular backups are presummated, and a disaster recovery plan is presumed to be in place in the event of data loss or system failure.

• Security: The database architecture takes data security into account. In order to safeguard sensitive data from unauthorised access and manipulation, security methods such as access controls, authentication protocols, encryption, and others are used.

• Platform and Technology: The database design presupposes that users will interact with the database using a certain database management system (such as MySQL) and underlying technology stack (such as a web application framework).