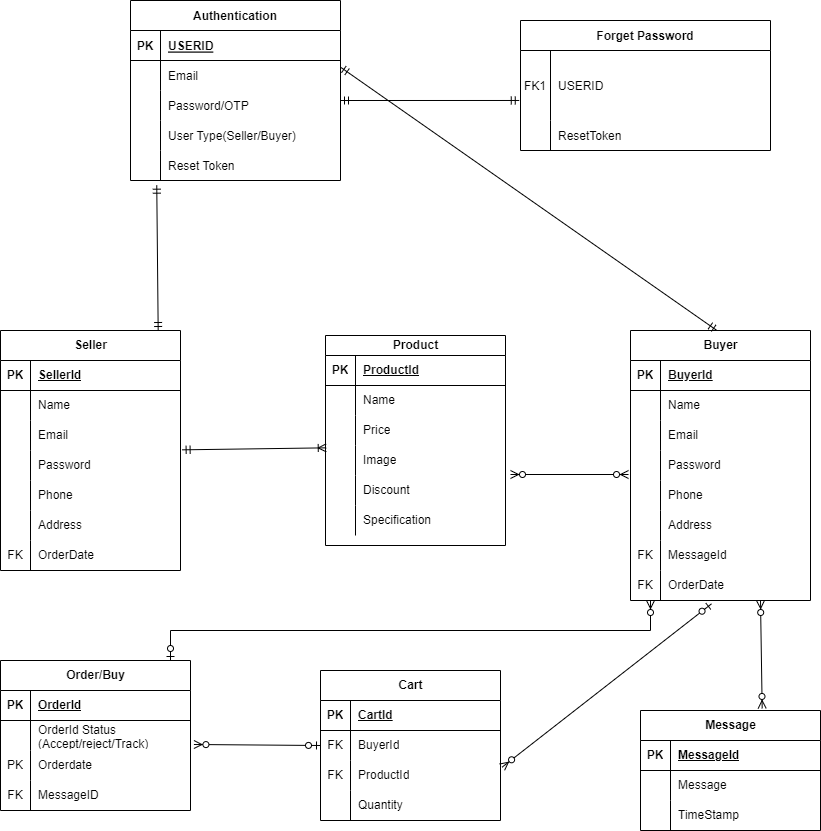
1. Physical Entity Relationship diagram of database.

Sol :



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

Sol :

* Searching performance is affected by a number of factors, including the size of the database, the complexity of the search query, and the hardware configuration of the server. In general, larger databases and more complex queries will take longer to execute. However, there are a number of things that can be done to improve searching performance, such as using indexes, partitioning the database, and optimizing the query.
* Replication is the process of copying data from one database to another. This can be done for a number of reasons, such as to improve performance, to provide redundancy, or to make data available to users in different locations. However, replication can also impact searching performance. If the replication process is not optimized, it can add latency to search queries.

There are a number of ways to handle replication in SQL for searching and reporting. One approach is to use a distributed query. This type of query allows you to run a single query against multiple databases. This can improve performance by reducing the number of times that the same data needs to be scanned.

Another approach is to use a materialized view. This is a snapshot of a database table that is stored in a separate database. Materialized views can be used to improve performance by caching frequently-accessed data.

Finally, you can also use a replication filter. This is a rule that determines which data is replicated to a subscriber database. By using a replication filter, you can ensure that only the data that is needed for searching and reporting is replicated.

The best approach for handling replication in SQL for searching and reporting will depend on the specific requirements of your application. However, the techniques described above can help to improve performance and ensure that your users have a good experience.

We should keep these checks for handling replication in SQL for searching and reporting:

* Use indexes on the tables that are used for searching and reporting.
* Partition the database to improve performance on large tables.
* Optimize the search queries to reduce the amount of data that needs to be scanned.
* Use a distributed query or materialized view to improve performance on queries that need to access data from multiple databases.
* Use a replication filter to ensure that only the data that is needed for searching and reporting is replicated.

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

Sol :

These are some of the major factors that are taken into consideration for performance in database design:

* Workload: The workload of the database is the most important factor that affects performance. The workload includes the types of queries that will be run, the frequency of those queries, and the size of the data that will be accessed.
* Data distribution: The distribution of data in the database can also have a significant impact on performance. If data is not distributed efficiently, it can lead to unnecessary I/O operations, which can slow down queries.
* Indexing: Indexes can be used to improve the performance of queries by providing a way to quickly locate the data that is needed. However, indexes can also add overhead, so it is important to use them judiciously.
* Query optimization: The way that queries are written can also have a significant impact on performance. Well-optimized queries can run much faster than poorly-optimized queries.
* Hardware configuration: The hardware configuration of the server can also affect performance. A database that is running on a slow server will not be as performant as a database that is running on a fast server.

In addition to these factors, there are a number of other factors that can affect performance, such as the database schema, the use of locking, and the amount of concurrency.

By carefully considering all of these factors, it is possible to design a database that will perform well under a variety of workloads.

4. Mention about Indexing, Normalization and Denormalization.

Sol :

* Indexing is a technique used to improve the performance of queries by providing a way to quickly locate the data that is needed. An index is a data structure that stores a subset of the data in a table, along with pointers to the actual data. When a query is run, the database engine can use the index to quickly locate the data that is needed, rather than having to scan the entire table.
* Normalization is a process of organizing data in a database to minimize redundancy and improve data integrity. Normalization is a multi-step process that involves dividing the data into multiple tables, and then defining the relationships between those tables. Normalization helps to improve performance by reducing the amount of data that needs to be scanned during a query. It also helps to improve data integrity by ensuring that data is not duplicated or inconsistent.
* Denormalization is the opposite of normalization. Denormalization involves adding redundant data to a database in order to improve performance. Denormalization is often used for tables that are frequently queried, and where the performance benefits of adding redundancy outweigh the drawbacks.

Below are some more details about each of these concepts:

* Indexing
  + There are two main types of indexes: clustered and non-clustered. A clustered index is a special type of index that actually orders the data in the table. A non-clustered index does not order the data in the table, but it does provide a way to quickly locate the data that is needed.
  + Indexes can be created on any column in a table, but they are most effective on columns that are frequently used in queries.
  + Indexes can improve the performance of queries by a factor of 10 or more.
* Normalization
  + There are five normal forms (1NF, 2NF, 3NF, BCNF, and 4NF). Each normal form adds a specific constraint to the data in a table.
  + 1NF ensures that each row in a table is unique.
  + 2NF ensures that each column in a table depends on the primary key.
  + 3NF ensures that each column in a table depends on the primary key, and not on any other column.
  + BCNF ensures that each column in a table depends on the primary key, and not on any other column that is not part of the primary key.
  + 4NF ensures that there are no multivalued dependencies in the table.
* Denormalization
  + There are two main reasons why denormalization is used: to improve performance and to simplify the data model.
  + Denormalization can improve performance by adding redundant data to a table. This can make it faster to find the data that is needed, especially for frequently-queried tables.
  + Denormalization can simplify the data model by reducing the number of tables that are needed. This can make it easier to understand and manage the data.

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

Sol :

The best approach will depend on the specific requirements of the application. However, some common approaches include:

* Vertical scaling involves increasing the capacity of a single database server. This can be done by adding more CPU cores, RAM, and storage. Vertical scaling is a good option for applications that have a predictable workload.
* Horizontal scaling involves adding more database servers to the cluster. This can be done by adding more nodes to a distributed database, or by adding more shards to a sharded database. Horizontal scaling is a good option for applications that have an unpredictable workload.
* Database partitioning involves dividing the data into multiple partitions, and then storing each partition on a separate database server. This can help to improve performance by distributing the load across multiple servers.
* Replication involves creating copies of the database on multiple servers. This can help to improve availability by providing a backup in case of a server failure.

When choosing a scaling strategy, it is important to consider the following factors:

* The size of the database. The larger the database, the more difficult it will be to scale.
* The workload of the application. If the application has a predictable workload, vertical scaling may be a good option. If the application has an unpredictable workload, horizontal scaling may be a better option.
* The budget. Scaling can be expensive, so it is important to consider the budget before making a decision.

By carefully considering these factors, it is possible to choose a scaling strategy that will meet the needs of the application and the budget.

We should keep that measures for handling scaling in database design:

* Use the appropriate data types for the data that will be stored in the database.
* Use normalization to reduce data redundancy.
* Partition the database to improve performance on large tables.
* Use indexes on the tables that are used frequently.
* Optimize the queries that are used frequently.
* Use a database that is designed for scalability.

6. Mention all the assumptions you are taking for solutions

Sol :

Below are the assumptions that I am taking for solutions in database design:

* The data is accurate and complete. This is a basic assumption that is necessary for any database design. If the data is not accurate or complete, then the database will not be able to provide accurate results.
* The data will be used in a predictable way. This assumption is important for determining the performance requirements of the database. If the data is used in an unpredictable way, then the database may not be able to handle the load.
* The data will grow at a predictable rate. This assumption is important for determining the scalability requirements of the database. If the data grows at an unpredictable rate, then the database may not be able to scale as needed.
* The database will be used by a predictable number of users. This assumption is important for determining the availability requirements of the database. If the database is used by a large number of users, then the database may need to be replicated or partitioned to improve availability.

These are just a few of the assumptions that I am taking for solutions in database design. The specific assumptions that I make will depend on the specific requirements of the application. However, these assumptions are a good starting point for any database design.

We also have to do these tasks for making assumptions in database design:

* Talk to the users of the database. The best way to understand the requirements of the database is to talk to the users who will be using it. The users can tell you what data they need, how they will use the data, and how much data they expect to store.
* Consider the future growth of the database. It is important to consider how the database will grow in the future. The database should be designed to handle the expected growth of the data.
* Use a variety of sources to gather information. In addition to talking to the users, you can also gather information from other sources, such as the business requirements document, the system architecture document, and the data dictionary.