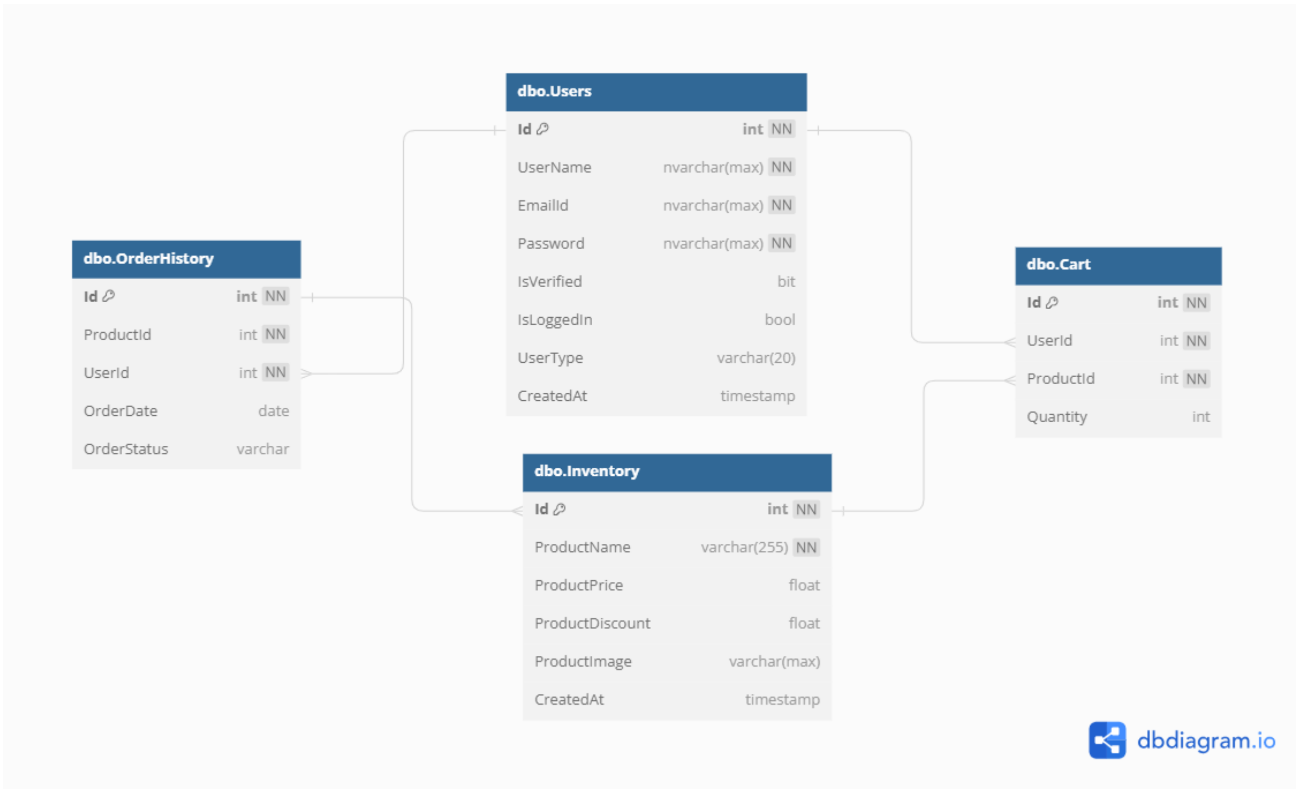
1. Physical Entity Relationship diagram of database.



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

Searching performance refers to the efficiency and speed at which a system can retrieve relevant information in response to search queries. It is a critical aspect of many applications, especially those involving large datasets and complex searches.

Here are some factors that influence searching performance:

Indexing: Creating appropriate indexes on columns frequently used in search queries can significantly speed up the search process. Indexes allow the database to locate data quickly, avoiding full table scans.

Query Optimization: Writing optimized queries with efficient JOINs, filtering conditions, and using appropriate search operators can improve performance. Avoiding complex queries and unnecessary operations is essential.

Caching: Implementing caching mechanisms can store frequently searched results in memory, reducing the need to execute queries repeatedly. This can improve response times and lower database load.

Full-Text Search: For text-heavy applications, using full-text search capabilities, such as those provided by specialized search engines like **Solr** or **Elasticsearch**, can enhance performance. These engines are designed to handle complex textual searches efficiently.

Partitioning and Sharding: For large databases, partitioning or sharding can distribute data across multiple servers, improving search performance by reducing the data volume per query.

Replication: Replication can help with searching performance by offloading read-intensive operations to read replicas, leaving the primary database more resources for write operations.

For Increasing searching in e-commerce applications I will use indexing on the Inventory table. Indexing is an important architecture styling to increase the performance of search. We implement indexing on the specific property of the table which is frequently used by users. There is one tool known as **Solr search engine** which is widely used in e-commerce websites.

Replication = duplication of data + synchronisation

For handling replication we have to create first. For creating replication I will use the master master method, because there will be lots of users who add their products to sell online.

Data replication in DBMS (distribution servers) can be carried out using a suitable replication scheme. The widely-adopted replication schemes are as follows:

1. Full data replication
2. Partial data replication
3. No replication

Full data replication

Full replication means that the complete database is replicated at every site of the distributed system. This scheme maximizes data availability and redundancy across a wide area network.

Partial data replication

Partial replication occurs when only certain fragments of the database are replicated based on the importance of data at each location. Here, the number of copies can range from one to the total number of nodes in the distributed system.

No replication

In this mode of replication, only one fragment exists on each site of the distributed system.

Create Read Replica: Set up read replicas to replicate data from the primary database. Read replicas can be located in different geographical regions to reduce latency and improve search performance for users in various locations.

Load Balancing: Implement load balancing to distribute read queries across multiple read replicas. A load balancer directs incoming read requests to available replicas, ensuring even distribution of the workload.

Failover Mechanism: Implement a failover mechanism to handle situations when a read replica becomes unavailable. The load balancer should automatically redirect traffic to a healthy replica.

Configuration and Monitoring: Configure replication settings and monitor replication lag to ensure that replicas are up-to-date and performing efficiently.

Routing Queries: Modify the application code to route read queries to the appropriate read replicas and write queries to the primary database. Use connection pooling to efficiently manage database connections.

Asynchronous Replication: Use asynchronous replication for read replicas to minimize the impact on the primary database's performance. Asynchronous replication allows replicas to lag slightly behind the primary database, ensuring write operations are not delayed.

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

Performance in the context of a website or software application refers to its ability to respond quickly and efficiently to user interactions, handle a large number of concurrent users, and deliver a smooth and responsive user experience. Several major factors are taken into consideration to assess and improve the performance of a website or application.

Key factors for performance:

1. **Response Time:** This is the time taken for the website or application to respond to a user's request. A lower response time indicates better performance, and it directly impacts user experience. Response time could be increased by using microservice architecture.

2. **Load Time:** Load time is the time it takes for a web page or application to fully load in a user's browser. Faster load times lead to higher user satisfaction and can positively influence search engine rankings.

3. **Replication:** Database replication effectively reduces the load on the primary server by dispersing it among other nodes in the distributed system, thereby improving network performance. By routing all read-operations to a replica database, IT administrators can save the primary server for write-operations that demand more processing power.

4. **Caching:** Implementing caching mechanisms helps store frequently accessed data temporarily, reducing the need to generate the same data repeatedly. Caching improves response times and reduces the load on backend servers.

5. **Optimized Code**: Well-optimized code reduces the execution time of operations, leading to faster responses. This includes efficient algorithms, database query optimization, and minimizing unnecessary computations.

6. **Database Performance:** The performance of database queries and operations is critical for applications heavily reliant on databases. Proper indexing, caching, and database tuning are essential for database performance.

7. **Content Delivery Network (CDN):** A content delivery network (CDN) is a network of interconnected servers that speeds up webpage loading for data-heavy applications. CDN can stand for content delivery network or content distribution network. When a user visits a website, data from that website's server has to travel across the internet to reach the user's computer. If the user is located far from that server, it will take a long time to load a large file, such as a video or website image. Instead, the website content is stored on CDN servers geographically closer to the users and reaches their computers much faster.

8. **Network Latency:** Reducing network latency, the time it takes for data to travel between the user's device and the server, is important for improving response times.

9. **Content Optimization**: Compressing images, minifying CSS and JavaScript, and using modern compression techniques reduce the size of assets sent to users, leading to faster load times.

To ensure better performance few points were taken care of in this project

1. Lower response time
2. Lower load time
3. Fast query
4. Optimized algorithms
5. Caching
6. Network latency
7. Replication

4. Mention about Indexing, Normalization and Denormalization.

Indexing:

Indexing is a data structure technique used to improve the speed of data retrieval in a database. Indexes are created on specific columns of a table, and they act as a quick reference to the location of data, allowing the database engine to locate the desired data efficiently without scanning the entire table. When a search query is executed on an indexed column, the database can use the index to quickly identify the relevant rows, resulting in faster query performance.

Key points about indexing:

- Indexes are typically used on columns frequently used in search and filtering operations.
- They trade off increased storage space and update overhead for improved query performance.
- Proper indexing can significantly enhance the speed of data retrieval for both read and search operations.
- Common types of indexes include B-tree indexes, hash indexes, and full-text indexes (for text search).

Normalization:

In the database design scope, Normalization is a database design technique that organizes tables in a manner that reduces redundancy and dependency of data by minimizing the insertion, deletion and update anomalies through eliminating the redundant data. Using the Normalization technique to design databases causes divide larger tables into smaller tables and links them using relationships. The purpose of Normalization is to eliminate redundant (useless) data and ensure data is stored logically.

Key points about normalization:

- It helps maintain data consistency and reduces data duplication, leading to better data integrity.
- Normalization avoids anomalies that could arise due to redundant data.
- However, normalization can sometimes lead to increased complexity in queries, requiring more JOIN operations to retrieve related data.
- It is used in OLTP systems, OLTP is optimized for transactional processing and real-time updates.

Denormalization:

Denormalization is the opposite of normalization. It involves intentionally introducing redundancy into the database by adding redundant data or combining multiple tables into a single table. The aim of denormalization is to improve read performance, simplify complex queries, and reduce the number of JOIN operations needed for certain queries. Denormalization is often used in read-heavy applications where optimizing query performance is a priority. By storing redundant data, the need for frequent JOIN operations across multiple tables can be reduced, resulting in faster data retrieval.

Key points about denormalization:

- It is a performance optimization technique for read-intensive applications.
- Denormalization can speed up read operations but might lead to increased storage requirements and update complexity.
- Care should be taken to maintain data consistency when denormalizing, as redundant data could introduce anomalies during updates.
- It is used in OLAP systems, OLAP is optimized for complex data analysis and reporting.

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

Handling scaling in an e-commerce website is essential to ensure it can handle increased traffic, transactions, and data without performance degradation or downtime. Here are some key strategies to help you scale an e-commerce website effectively:

1. **Use Cloud Hosting**: Consider using cloud hosting services like Amazon Web Services (AWS), Google Cloud Platform (GCP), or Microsoft Azure. Cloud hosting allows for easy scalability, as you can increase server resources on-demand based on traffic spikes.

2. **Optimize Database Performance**: Optimize database queries, use caching mechanisms, and consider database sharding or replication to distribute the database load effectively.

3. **Load Balancing**: Implement load balancing to distribute incoming traffic across multiple servers. This ensures the even distribution of requests and prevents any single server from being overloaded.

4. **Caching:** Utilize caching mechanisms to store frequently accessed data and reduce the need to regenerate it with each request. Use caching plugins or technologies like Redis or Memcached.

5. **Horizontal Scaling:** Instead of increasing the resources of a single server, consider adding more servers to handle the increasing load. This is known as horizontal scaling and is an effective way to handle higher traffic.

6. **Optimize Code and Resources:** Optimize your code, images, and other assets to reduce their size and improve loading speed. Minimize HTTP requests and remove any unnecessary plugins or scripts.

7. **Auto-scaling:** Implement auto-scaling mechanisms that automatically add or remove resources based on predefined conditions, such as CPU utilization or network traffic.

8. **Distributed Architecture:** Consider breaking down your application into microservices and using a distributed architecture. This enables independent scaling of different components based on their individual needs.

9. **Plan for Seasonal Peaks**: If your e-commerce website experiences seasonal spikes in traffic (e.g., during holidays or special events), plan ahead and allocate additional resources to handle the increased load.

Scalability can be handled in multiple ways in this project.

1. By using distributed system architecture
2. By using the Load balancing technique
3. By moving to cloud hosting that helps in in-demand auto-scaling
4. By Using Caching
5. By Using Fast Searching in the database
6. By Horizontal scaling

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

For Authentication and Authorization

Authentication and authorization work together to ensure the security and integrity of a system. Authentication verifies the identity of users, while authorization governs what those authenticated users are allowed to do within the system. Properly implementing both authentication and authorization measures is critical for protecting sensitive information and maintaining control over system resources.

1. I will generate a JWT token for logged in users.
2. Generated JWT token will be stored at client side.
3. Verification email will be sent to the new registered users.
4. User’s email id and password will be required for login
5. If user forget their password, forgot password functionality will be triggered
6. If a user is present in the table then I will send the password to the user's email id.

For Inventory Table

In order to build an e-commerce application Inventory table is an essential part of the database. I will create an Inventory table with the property of product image, name, price, discount and specification. After successfully creating the table there I will perform many operations on this table.

Operation on Inventory Table

1. Sellers can add products in the inventory table with the seller Id.
2. All the data of the inventory table will be visible to all types of users.
3. If users are a type of seller, they can delete their inserted data from the inventory table.
4. When a seller adds a new product into inventory push notification will be sent to the buyer and email notification will be sent to all the users present in the users table.

For Cart Table

Like any e-commerce web application, if a user wants to buy a product he/she adds that item to the cart and goes to the product page If the user wants to buy other items. The Cart table is a crucial component that helps manage the items that users have added to their shopping cart before proceeding to the checkout and making a purchase. The Cart table typically stores temporary data associated with each user's shopping session.

To add items to cart I have created the cart table with the user_id and product_id property in the database.

Operations on Cart Table

1. Only logged in users can add items to the cart.
2. Users id and product id will be stored in the cart table.
3. Logged in users can add multiple items to the cart.
4. Users can also delete the data from cart, if they don't want to buy them.

For Order History Table

An "Order History" table in an e-commerce application is used to store historical information related to completed orders. It keeps a record of past orders placed by users, including details such as order number, customer information, shipping details, payment status, and more.

Below is some operations on the Order History table:

1. Logged users will be able to order the product.
2. Id of logged in users will be stored in the order history table.
3. Id of the product will also be stored in the Order History table.
4. When users click to buy a product then one email will be sent to the buyer and one email will be sent to the seller of that product.
5. Status property in the Order History will manage the status of order, If the seller refuses to sell the project status will become rejected and if he accepted the status would be accepted and ready to ship.

For Scaling

To build this e-commerce application I will use microservices architecture. And the design of the database will be distributed. Microservice architecture helps to scale up the application at any point of time. Microservices also increase the response time of the application and it handles errors. If any service fails at any time then this failure does not impact the other running services.