Q1. Physical Entity Relationship diagram of database.

A diagram of a computer

Description automatically generated with medium confidence

Q2. Explain about searching performance. How will you handle replication in SQL for searching & Reporting?

Solution2. For searching of products, we are using Elasticsearch, as we are dumping our product database to Elasticsearch the retrieval of data should be fast and the performance will increase. The following reasons why I am using Elasticsearch for searching of items are: -

**Optimized Retrieval Speed:** Elasticsearch employs inverted indexing, facilitating swift retrieval of items. This approach surpasses traditional methods such as cluster indexing in MySQL databases. Given the high read-to-write ratio prevalent in our inventory, Elasticsearch's efficiency becomes especially relevant. Additionally, the ability to shard indexes across multiple nodes in a cluster ensures rapid data retrieval, particularly useful for geographical or regional searches.

**Scalability and Sharding:** Elasticsearch supports sharding of indexes across nodes, leading to improved scalability. This feature is vital for accommodating the dynamic growth of our product database. By distributing the workload across multiple nodes, Elasticsearch ensures that the search performance scales seamlessly with the expansion of our inventory.

**Fault Tolerance:** The inherent fault tolerance of Elasticsearch adds a layer of robustness to our system. Each shard in Elasticsearch maintains its own replica. In the event of a primary shard failure, the replica takes over seamlessly, ensuring uninterrupted operations. This redundancy significantly reduces the risk of data loss or service disruptions due to hardware failures or other unforeseen issues.

Q3. Explain what major factors are taken into consideration for performance.

Solution3. Major factors that are taken into consideration for performance are:

**1.** **Load** **Balancer**- As there is going to be a replica of our databases to different servers so load balancer will distribute the traffic accordingly using some hashing mechanism based on regions or demands which give user high availability and performance.

**2.** **Sharding**- Distribution of data over servers give us high availability and performance in every region.

**3. Elasticsearch**- Elasticsearch is used for searching of products.

**4. Normalization**- Services in which there is high write: read ratio normalization is used to increase performance for example user authentication and authorization service is write intensive therefore normalization is used to reduce redundancy as if there is a replication of database over servers there is going to be less latency as data is not redundant.

**5. Denormalization**- Services which are read intensive like inventory service there is more read: write ratio and there is some complex joins in inventory such as self-join of category therefore NoSQL database is used to be safe from costly complex joins operations and to increase performance.

Q4. Mention about Indexing, Normalization and Denormalization.

Solution 4. **Indexing**: Elasticsearch employs inverted indexing, enabling rapid data retrieval. This mapping of terms to document locations enhances search performance.

* In SQL, indexing on the order date in the Order table accelerates queries related to order dates.

**Normalization**: Normalization is crucial for relational databases, reducing redundancy and enhancing data integrity.

* For authentication and authorization, separating the address into its own table exemplifies normalization, preventing data duplication and ensuring user information consistency.
* In order management, normalizing to the 3rd normal form eliminates partial and transitive dependencies, maintaining data consistency and integrity.

**Denormalization**: Denormalization, applied in read-intensive scenarios, simplifies data retrieval.

* In the inventory module using MongoDB, denormalization involves storing hierarchical structures like categories and SKUs together, improving retrieval speed, and avoiding null values for certain fields to optimize storage efficiency.

Q5. How will you handle scaling, if required at any point of time.

Solution5. Scaling is handled using different ways:

**Vertical** **Scaling:** Initially, vertical scaling can be employed by upgrading the existing server's hardware, such as increasing CPU, RAM, or storage capacity.

**Auto Scaling:** Utilize auto-scaling features provided by cloud platforms to dynamically adjust the number of servers based on real-time demand. This ensures efficient resource utilization and cost-effectiveness.

**Sharding:** To address scalability concerns with the database, sharding can be implemented. Sharding involves distributing the database into smaller, more manageable parts (shards) across multiple servers.

Q6. Mention all the assumptions you are taking for solutions.

Solution 6.

* For the implementation of OTP (One-Time Password) authentication, the approach involves storing the OTP temporarily in memory and assuming automatic deletion once it is successfully used. There is no provision for a separate database to persistently store OTPs, implying that monitoring or manual intervention is not required for the deletion process.

This method relies on an in-memory storage mechanism where the OTP is generated, stored temporarily, and then removed automatically after a single-use scenario. The absence of a dedicated database implies a streamlined and efficient process, avoiding the need for continuous monitoring to clear or manage used OTPs. The temporary storage of OTPs in memory ensures a secure and convenient means of authentication while minimizing the need for additional database management for OTP storage.