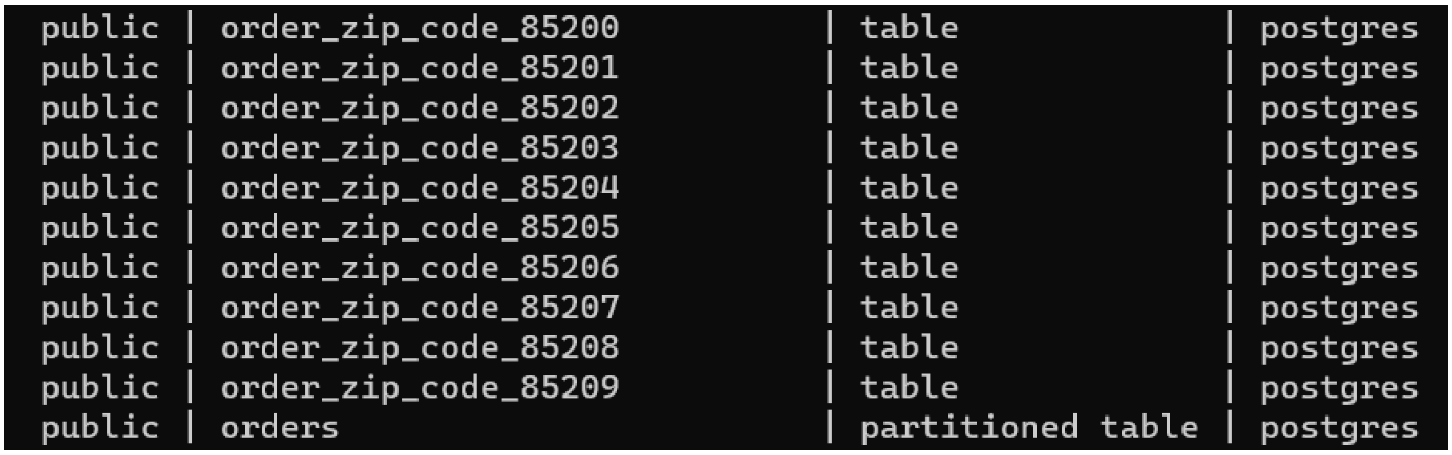
Paseo Posse – Hyperlocal Delivery System

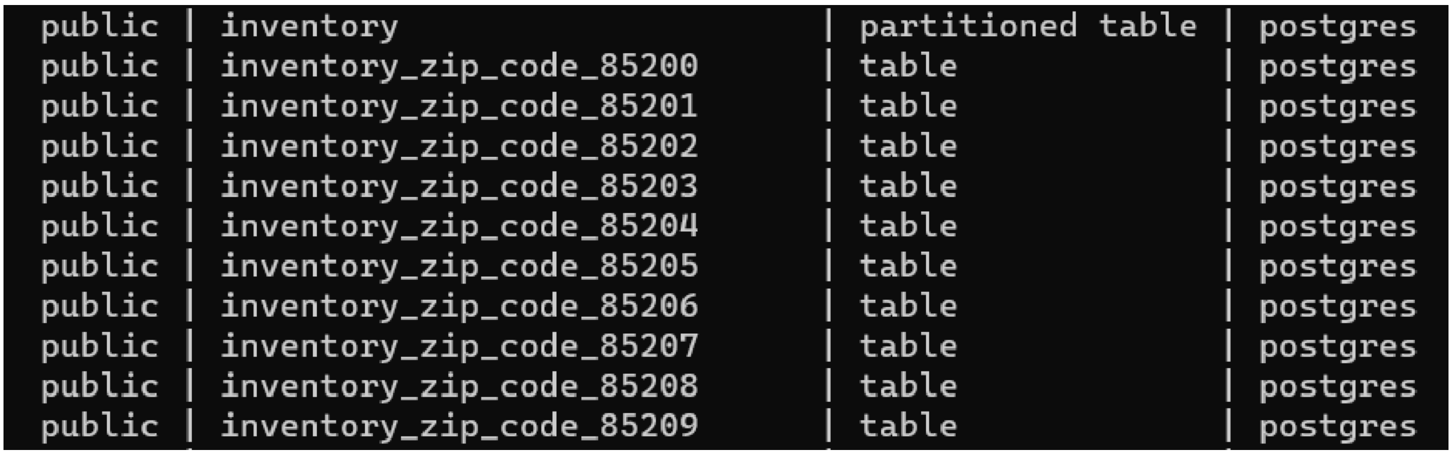
In our project, we have implemented concepts of Horizontal Fragmentation and Replication. Horizontal fragmentation refers to the division of the database tables based on certain criteria or attributes. Each fragment represents a subset of the data that satisfies specific conditions. Horizontal fragmentation can be employed to optimize data distribution, improve query performance, and enhance scalability. Let's discuss how horizontal fragmentation is applied to specific tables in the project:

The Orders table includes information about customer orders, the table is horizontally fragmented based on the geographical location of ZIP codes. Each fragment corresponds to orders from customers in a specific ZIP code. This fragmentation strategy ensures that orders are stored in partitions corresponding to their delivery locations.



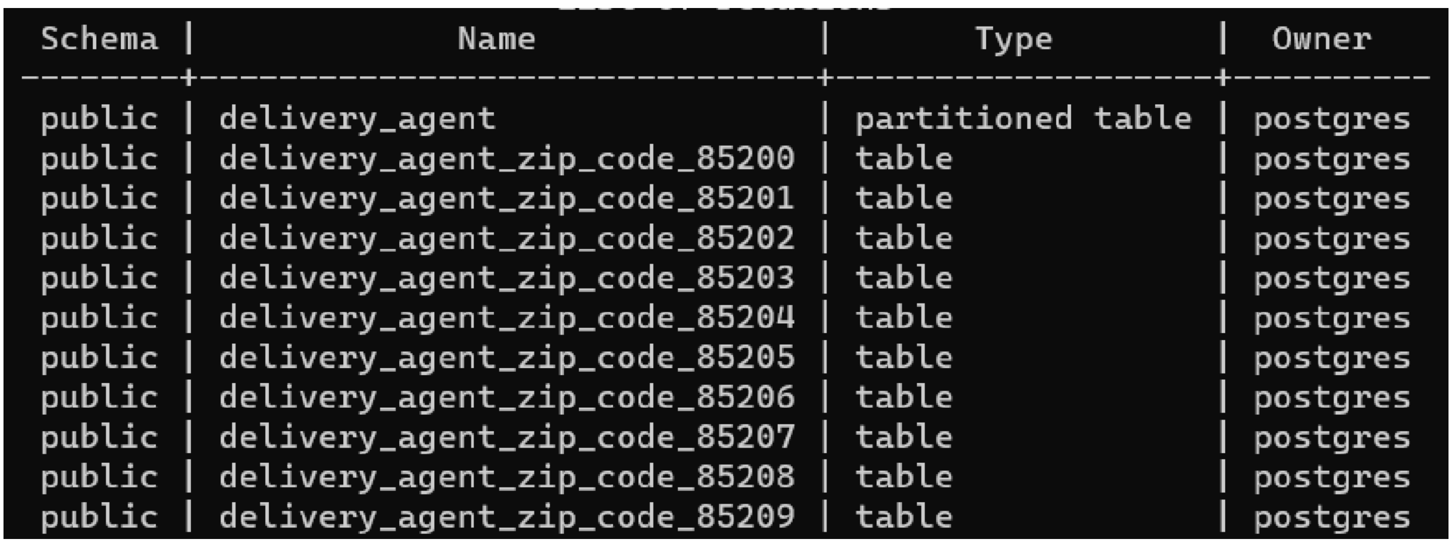
```
 public | order_zip_code_85200          | table             | postgres
 public | order_zip_code_85201          | table             | postgres
 public | order_zip_code_85202          | table             | postgres
 public | order_zip_code_85203          | table             | postgres
 public | order_zip_code_85204          | table             | postgres
 public | order_zip_code_85205          | table             | postgres
 public | order_zip_code_85206          | table             | postgres
 public | order_zip_code_85207          | table             | postgres
 public | order_zip_code_85208          | table             | postgres
 public | order_zip_code_85209          | table             | postgres
 public | orders                        | partitioned table | postgres
```

For the Inventory table, which contains information about available stock in different regions, horizontal fragmentation is based on ZIP codes as well. Each fragment represents the inventory for a specific ZIP code, allowing for localized tracking and efficient retrieval.



```
 public | inventory                     | partitioned table | postgres
 public | inventory_zip_code_85200      | table             | postgres
 public | inventory_zip_code_85201      | table             | postgres
 public | inventory_zip_code_85202      | table             | postgres
 public | inventory_zip_code_85203      | table             | postgres
 public | inventory_zip_code_85204      | table             | postgres
 public | inventory_zip_code_85205      | table             | postgres
 public | inventory_zip_code_85206      | table             | postgres
 public | inventory_zip_code_85207      | table             | postgres
 public | inventory_zip_code_85208      | table             | postgres
 public | inventory_zip_code_85209      | table             | postgres
```

The DeliveryAgent table, which assigns delivery agents to specific regions, also benefits from horizontal fragmentation based on ZIP codes. Each fragment represents the delivery agents assigned to a particular ZIP code, enabling efficient tracking and assignment.



```
 Schema |               Name               |       Type        |  Owner
--------+----------------------------------+-------------------+----------
 public | delivery_agent                   | partitioned table | postgres
 public | delivery_agent_zip_code_85200    | table             | postgres
 public | delivery_agent_zip_code_85201    | table             | postgres
 public | delivery_agent_zip_code_85202    | table             | postgres
 public | delivery_agent_zip_code_85203    | table             | postgres
 public | delivery_agent_zip_code_85204    | table             | postgres
 public | delivery_agent_zip_code_85205    | table             | postgres
 public | delivery_agent_zip_code_85206    | table             | postgres
 public | delivery_agent_zip_code_85207    | table             | postgres
 public | delivery_agent_zip_code_85208    | table             | postgres
 public | delivery_agent_zip_code_85209    | table             | postgres
```

Benefits of Horizontal Fragmentation:

Improved Query Performance:

Querying a horizontally fragmented table for a specific region is more efficient as it involves scanning a smaller subset of the data.

Localized Data Management:

Data related to a specific region is stored together, facilitating localized management and reducing the need to access data from distant locations.

Efficient Resource Utilization:

Each fragment can be stored on servers located close to the corresponding region, optimizing resource utilization, and reducing network latency.

Scalability:

New fragments can be added as the service expands to new regions, allowing the system to scale horizontally without affecting the entire dataset.

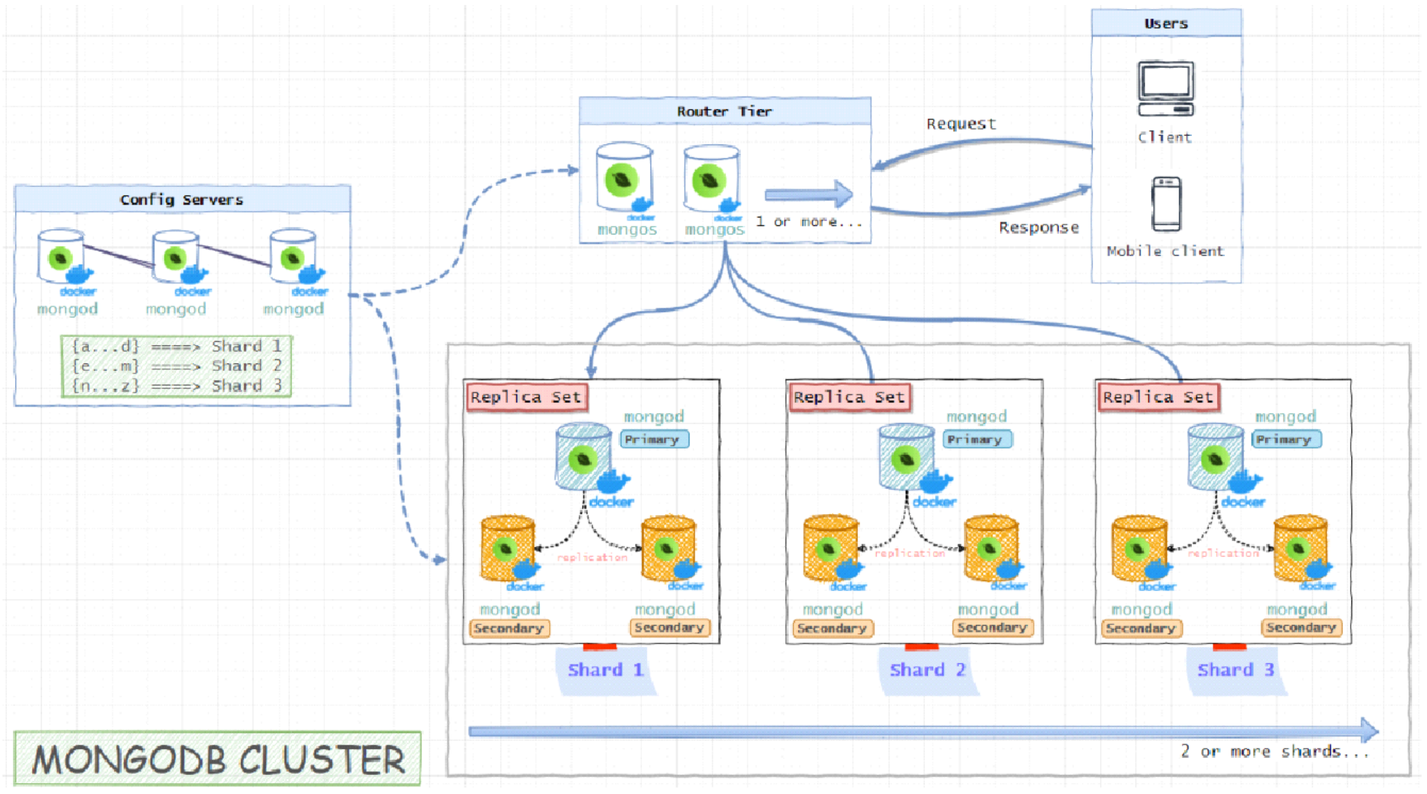
Concurrency Control:

Horizontal fragmentation can enhance concurrency control, as operations on different fragments can be executed independently, reducing contention.

Replication:

In MongoDB, replication is used to provide high availability and fault tolerance by maintaining multiple copies of data across different servers. In a master-slave replication setup, one node (the master) accepts write operations, while the other nodes (the slaves) replicate data from the master and serve read operations. This setup enhances data availability and provides fault tolerance.

In our case, we have a MongoDB sharded cluster with a configuration server replica set and three shards, each consisting of a three-member PSS (Primary-Secondary-Secondary) replica set. Additionally, we have two routers (mongos) for query routing.



Users
Client
Mobile client
Router Tier
mongos
mongos
1 or more...
Request
Response
Config Servers
mongod
mongod
mongod
{a...d} ====> Shard 1
{e...m} ====> Shard 2
{n...z} ====> Shard 3
Replica Set
mongod
Primary
mongod
Secondary
mongod
Secondary
replication
Shard 1
Shard 2
Shard 3
2 or more shards...
MONGODB CLUSTER