RMIT University Vietnam

School of Science, Engineering and Technology

(SSET)

ISYS2099 – Database Applications

Lazada Project

Lecturer: Dr. Tri Dang Tran

Group 4

Tran Phuong Anh – s3914138

Nguyen Hoang Khang – s3802040

Nguyen Dinh Dang Nguyen – s3759957

Database Design

Based on the requirement, there are 2 types of databases: relational and non-relational. The relational database used in this project is MySQL and non-relational database of choice is MongoDB. There are 8 tables in the MySQL database: Customers, Hubs, Orders, Products, Roles, Shippers, Users, and Vendors. First, the Customer entity has a “one-to-one” relationship with the Order entity. Product has “one-to-one” relationship with Order. For every user including Shipper, Vendor, and Customer, a record will be created in the User table, apart from the Shippers, Vendors and Customers table. A Role can be assigned to many users and a User can only have one role. There are three roles: Vendor, Customer and Shipper. A hub can be assigned to many shippers; hence, multiple shippers can have the same list of orders.

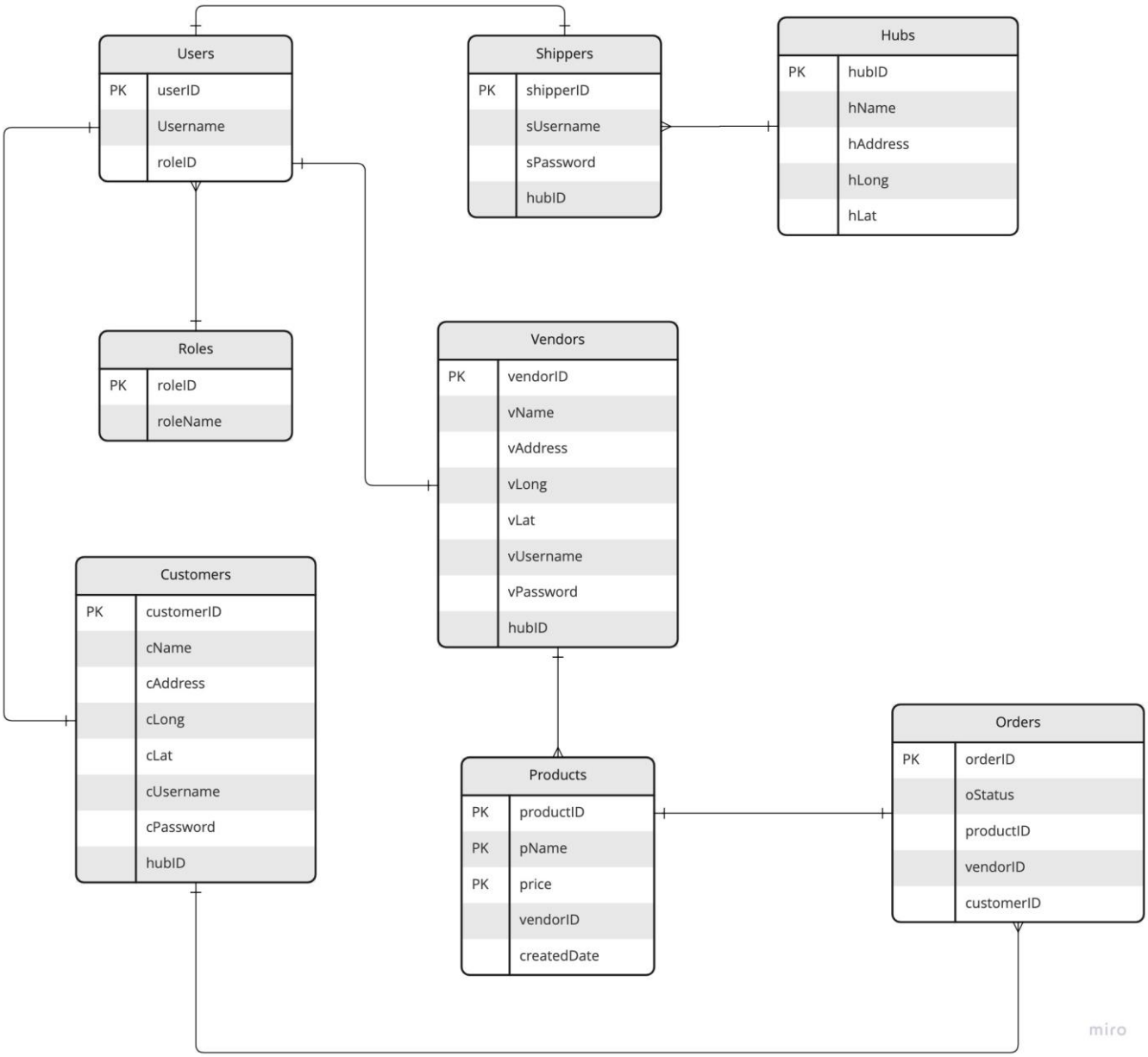


Figure 1: Entity Relationship Diagram

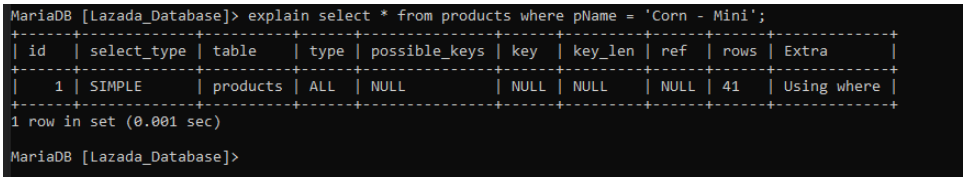
Database Performance and Trade-offs

To achieve high performance, the design choices are:

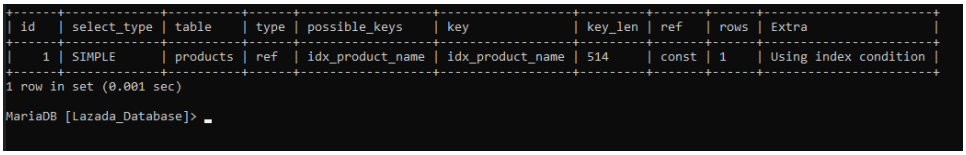
- Indexing: Create Index on “product name” (pName column) to do the search quickly. The tradeoff is we use more storage to store indexes.
- Partitioning: Create partition based on the range of price.
- Query optimization: We avoid expensive query operations such as Joins.

Performance analysis:

No index

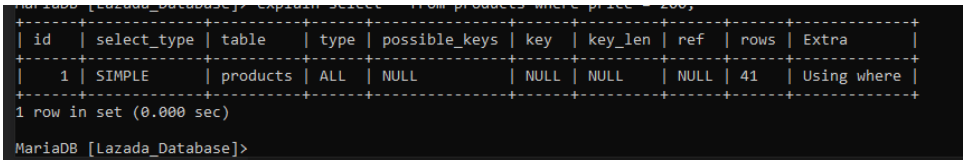


Index solely



By indexing on pName, it increases the performance when searching for name of the product.

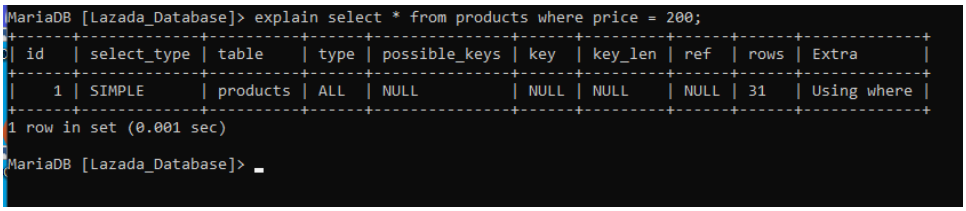
No Partition



```
+------+-------------+----------+------+---------------+------+---------+------+------+-------------+
| id   | select_type | table    | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+------+-------------+----------+------+---------------+------+---------+------+------+-------------+
|    1 | SIMPLE      | products | ALL  | NULL          | NULL | NULL    | NULL | 41   | Using where |
+------+-------------+----------+------+---------------+------+---------+------+------+-------------+
1 row in set (0.000 sec)

MariaDB [Lazada_Database]>
```

Partition only



```
MariaDB [Lazada_Database]> explain select * from products where price = 200;
+------+-------------+----------+------+---------------+------+---------+------+------+-------------+
| id   | select_type | table    | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+------+-------------+----------+------+---------------+------+---------+------+------+-------------+
|    1 | SIMPLE      | products | ALL  | NULL          | NULL | NULL    | NULL | 31   | Using where |
+------+-------------+----------+------+---------------+------+---------+------+------+-------------+
1 row in set (0.001 sec)

MariaDB [Lazada_Database]>
```

As we set partition by range of price, there is a significant change in performance where value Price = 200 is searched for (from 41 rows to 31 rows).

Tradeoff between transaction isolation level and database performance (the higher the level, the worse the performance)

Trade-off

We have traded off, given away memory in return for time execution.

Data integrity and Consistency

As for data integrity and consistency, we apply database triggers and functions for automatic assigning of Distribution Hubs to Vendors and Customers.

First, we created a database function to calculate distance in kilometers based on longitude and latitude with the Vincenty formula [1]. Then, we call the function in a trigger before insertion of a new Vendor to assign the Hub to the Vendor.

As for the assignment of Hub to a Customer, respective Hub ID is stored directly in each Customer in a “one-to-one" relationship. In essence, every time a new customer account is created, a trigger will be called before insertion to assign the Hub ID to the newly created order. A mathematical function is set up to determine the nearest Hub ID to a customer given input of that customer’s location.

In terms of concurrency handling, we approach the problem with pessimistic locking by creating row locks on Repeatable Reads Isolation Level. To implement it, we use stored procedures to store a transaction with:

- **For update (Exclusive) lock** in case 2 shippers are updating 1 specific order simultaneously. This way, the only way the second shipper could update is to wait until the first shipper finished updating completely.
- **For share (Shared) lock** in case 1 customer is making a purchase of a product (which, new order is inserted), vendor can’t update that specific product at the same time. In this part, a vendor cannot update its product unless the insertion of new order has been completed.

Data security and privacy

We apply standard database security and privacy procedures. First, we use prepared statements, implement input validation, and escape special characters in our PHP codebase to prevent SQL injection. We also apply Role-Based Access Control (RBAC) and Views at the database level and password hashing to prevent loss of confidentiality and loss of integrity.

For RBAC, there are 6 users (lazada_admin, lazada_customer, lazada_vendor, lazada_shipper, lazada_guest, lazada_auth), and 5 roles (Customer, Vendor, Shipper, Guest, Auth). The user lazada_admin has full access to the entire database and its credentials should only be accessible by a database administrator. The other 5 users are granted corresponding roles with permissions limited to their user features. The Guest and Auth roles are for user authentication and registration. Guests can insert new data to table Customers, Shippers, Vendors but cannot select. Auth can select data from Customers, Shippers, Vendors but cannot insert. This approach is to prevent Guest from viewing users' passwords and Auth to self-register. There are also 3 views (customers_hidepass, vendors_hidepass, shippers_hidepass) for Customer, Vendor, Shipper roles to select users' information without compromising their passwords.

References

[1] Vincenty, "Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations." [Online]. Available: https://www.ngs.noaa.gov/PUBS_LIB/inverse.pdf. [Accessed: 10-Sep-2022].