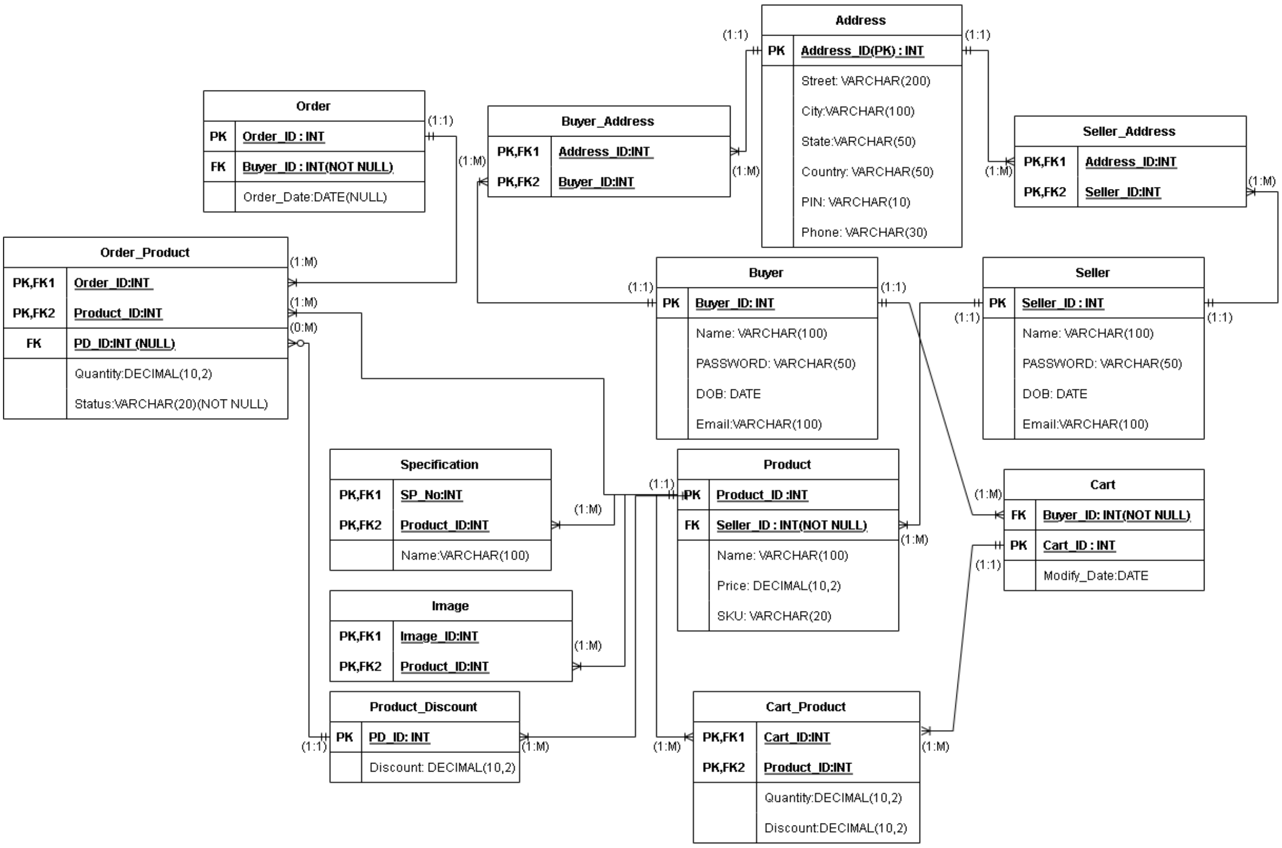
Table of Contents

Physical ER diagram of database



Assumptions for the solution

RDBMS Tables

1. Buyer
   i) Buyer will just have name and DOB.
   ii) Each buyer will have a unique id.
   iii) Each buyer can have multiple addresses.

2. Seller
   i) Seller will just have name and DOB.
   ii) Each seller will have a unique id.
   iii) Each seller can have multiple addresses.
   iv) A seller can sell more than one products.

3. Address
   i) Each address has a unique address_id.
   ii) All the addresses for a buyer/seller will be stored in this table.

4. Buyer_Address
   i) Buyer_address is used for mapping a buyer with an address.
   ii) A combination of buyer_id and address_id will be used as a composite primary key.

5. Seller_Address
   i) Seller_address is used for mapping a buyer with an address.
   ii) A combination of seller_id and address_id will be used as a composite primary key.

6. Product
   i) Each product will have a unique product_id.
   ii) Each product will be sold by only a single seller.
   iii) A product can have more than one specifications.
   iv) A product can have more than one images.
   v) A product can have only a single discount.

7. Specification
   i) Each specification has a unique specification id.
   ii) Each specification will be associated to a single product.
   iii) The specifications will be stored as a string.

8. Image
   i) Each image has a unique image id.
   ii) Each image will be associated to a single product.
   iii) The image will be stored as a url.

9. Product_Discount
    i) Each product_discount has a unique discount id.
    ii) Each discount will be associated to a single product.
    iii) The discount will be stored as a number.

10. Cart
    i) Each cart will have a single buyer.
    ii) Each cart will have a unique cart id.
    iii) Each cart can have multiple products.

11. Cart_Product
    i) Each cart_product will have a unique combination of cart_id and product_id.
    ii) The discount will be associated with each cart_product.
    iii) Each cart_product will have a single product.

12. Order
    i) Each order will have a unique order_id.
    ii) Each order will have a unique buyer.
    iii) An order can have multiple order products.

13. Order_Product
    i) Each order_product will have a unique combination of order_id and product_id.
    ii) The discount will be associated with each order_product.
    iii) Each order_product will have a single product.
    iv) The seller can either accept/reject this order_product.

No-SQL Tables

We will have a Order_History No-SQL database which will have a structure like below example:

```
{
   "_id" : 1,
   "order_id" : 123,
   "buyer_id": 456,
   "seller_id": 345,
   "order_date": new ISODate("2020-05-18T14:10:30Z") ,


   "products": [
     {
       "product_id": 678,
       "name": "Book Let us java",
       "sku": "12345678",
       "price": 120.50,
            “quantity”:3,
       "discount": 12.50,
            “image”:”www.test.com/abc.png”
     },
     {
       "product_id": 234,
       "name": "Pen",
       "sku": "34765672",
       "price": 122.50,
            “quantity”:2,
       "discount": 0,
            “image”:”www.test.com/snh.png”
     }    ]
 }
```

Searching performance

1. We will be creating indexes on the basis of columns that are frequently used.
2. We will also be using replicated databases which will be available in multiple zones and regions. The products table will be subdivided into a master table which will allow reads/writes and multiple replicas which will just allow reads. So the users searching for a product will get the result from their nearby regions/zones.
3. We will also create indexes using product names as product_name will be used for searching.

Where should we keep replication in NO-SQL or SQL for searching & Reporting?

We can keep the replication in SQL/No-SQL as per the requirements. In the current scenario, we can use SQL for searching as we will have several read replicas as we will get the answer/result in better time.

For reporting, we have opted for NO-SQL as all the data will be stored at a single place/document. We will be storing order documents and each order document will have all the data related to that particular order.

Majors taken for performance, archiving and purging

Performance

1. We will be creating indexes on the basis of columns that are frequently used.
2. Read replicas will be used for reading data from the database.

Archiving

1. We will be using separate NoSql database for storing order_history and the orders will be moved in this document from order table after one year.
2. There will be jobs responsible for archiving this data.
3. This order_history NoSQL database will be storing all the data related to an order at a single place which will result in a better performance.

Purging

1. There will be jobs responsible for purging this data.
2. Data older than 3 years will be deleted from the Order_history NoSql database.

Security consideration

1. Sensitive data will be encrypted.
2. Proper firewalls will be configured.
3. We will be using Windows authentication mode as it is more secured.
4. Least privileges will be given to the users(Minimal Privilege Principle).
5. Password policies and password expiration policies will be applied.
6. Proper audits will be performed.
7. DML, DDL and logon triggers will be applied.
8. Proper backups will be created for the database.
9. We will be using solid SQL monitoring tool to scan the processes of database application and monitor the changes in database settings.
10. All the backups will be properly secured.
11. The database will be protected for SQL injections.