Alejandro Rebolloso

Oluchi Obinna

Professor Bollampally

ITSE 1346 Database Theory

8 December 2024

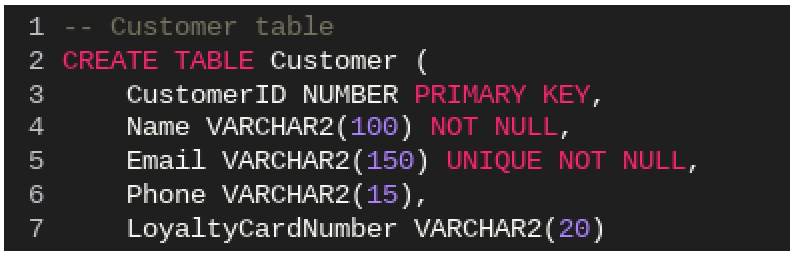
Project 2 Documentation

The objective was to translate the logical model we created into a physical database by writing and executing DDL and DML scripts.

DDL Script Creation

The purpose of the DDL scripts is to create tables for each entity in our ERD. To create the table, the function CREATE TABLE was used for each entity in the logical model. In this script, we created several tables for an e-commerce or retail database system. The Customer table stores customer details such as their name, email, phone, and loyalty card number. The Product table holds product information like product ID, name, description, price, and stock quantity, with constraints to ensure the price is positive and stock is non-negative. The Orders table records order details, including order ID, customer ID, order date, and total amount, with a foreign key to link customers and orders.

Example:



```
-- Customer table
CREATE TABLE Customer (
    CustomerID NUMBER PRIMARY KEY,
    Name VARCHAR2(100) NOT NULL,
    Email VARCHAR2(150) UNIQUE NOT NULL,
    Phone VARCHAR2(15),
    LoyaltyCardNumber VARCHAR2(20)
```

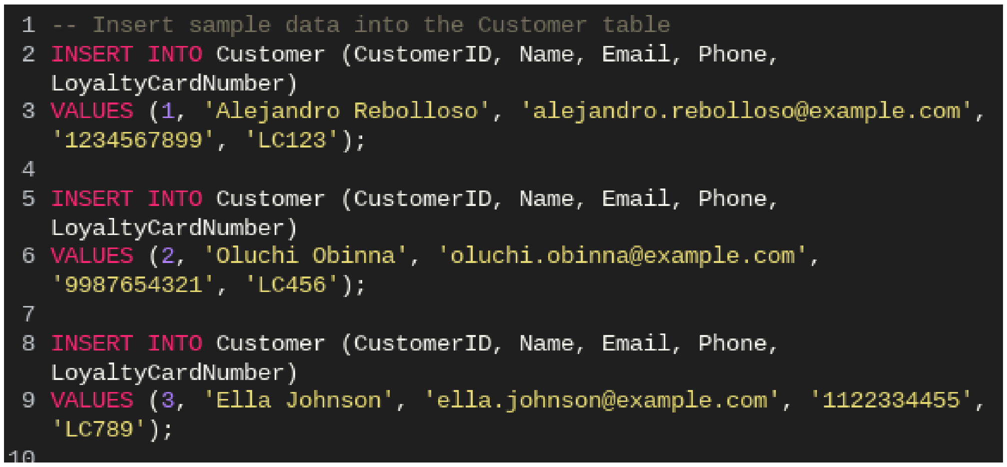
A simple order was followed for the rest of the entities to keep the scripts clean and legible. The order goes: Primary Key, followed by mandatory attributes, and lastly optional attributes.

The OrderDetails table keeps track of which products are part of each order, including quantity and unit price, with foreign keys referencing the Orders and Product tables. The Payment table stores payment details for each order, including payment type and amount, with a foreign key to link payments to orders. The Discount table manages discount details, specifying a minimum purchase and a discount percentage. Finally, the ItemPriceHistory table records changes in product prices over time, with foreign keys linking it to the Product table.

DML Script Creation

DML statements were used to populate the tables with data to demonstrate the basic functions. The DML Script samples data into the tables from our database, including customer, product, order, payment, discount, and item price history information. We put ourselves and an extra as the three customers are added with their personal details. To fill out the fields, orders are created for the customers, linking them to specific products, quantities, and total amounts, and payments are recorded for each order.

In this example, the INSERT statement was used to populate the table for each entity.



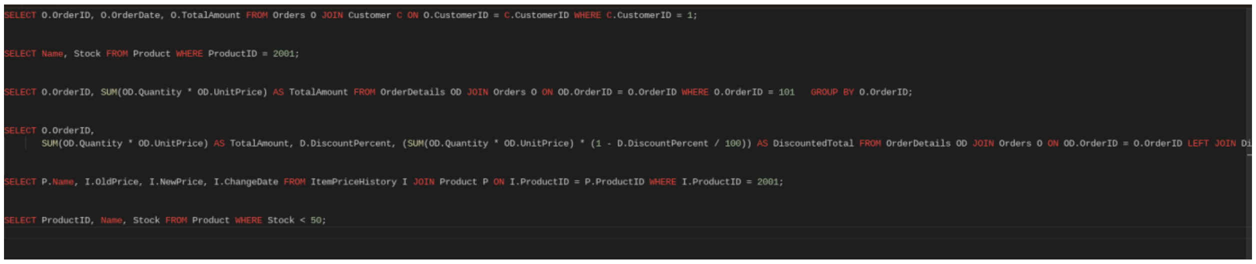
```sql
-- Insert sample data into the Customer table
INSERT INTO Customer (CustomerID, Name, Email, Phone,
LoyaltyCardNumber)
VALUES (1, 'Alejandro Rebolloso', 'alejandro.rebolloso@example.com',
'1234567899', 'LC123');

INSERT INTO Customer (CustomerID, Name, Email, Phone,
LoyaltyCardNumber)
VALUES (2, 'Oluchi Obinna', 'oluchi.obinna@example.com',
'9987654321', 'LC456');

INSERT INTO Customer (CustomerID, Name, Email, Phone,
LoyaltyCardNumber)
VALUES (3, 'Ella Johnson', 'ella.johnson@example.com', '1122334455',
'LC789');
```

SQL Queries

The objective is to write SQL queries for retrieving meaningful data from the database, such as fetching all orders for a customer, checking product stock availability, and calculating order totals or applying discounts. Each query should be executed and validated to ensure it returns the expected results.



```sql
SELECT O.OrderID, O.OrderDate, O.TotalAmount FROM Orders O JOIN Customer C ON O.CustomerID = C.CustomerID WHERE C.CustomerID = 1;

SELECT Name, Stock FROM Product WHERE ProductID = 2001;

SELECT O.OrderID, SUM(OD.Quantity * OD.UnitPrice) AS TotalAmount FROM OrderDetails OD JOIN Orders O ON OD.OrderID = O.OrderID WHERE O.OrderID = 101   GROUP BY O.OrderID;

SELECT O.OrderID,
    SUM(OD.Quantity * OD.UnitPrice) AS TotalAmount, D.DiscountPercent, (SUM(OD.Quantity * OD.UnitPrice) * (1 - D.DiscountPercent / 100)) AS DiscountedTotal FROM OrderDetails OD JOIN Orders O ON OD.OrderID = O.OrderID LEFT JOIN Di

SELECT P.Name, I.OldPrice, I.NewPrice, I.ChangeDate FROM ItemPriceHistory I JOIN Product P ON I.ProductID = P.ProductID WHERE I.ProductID = 2001;

SELECT ProductID, Name, Stock FROM Product WHERE Stock < 50;
```

To achieve this, we can use various SQL queries to retrieve and analyze data from the database. For example, we can retrieve all orders for a specific customer, such as CustomerID 1, by joining the Orders and Customer tables to display relevant order details. We can check the stock availability of a specific product (ProductID 2001) or calculate the total amount for a particular order, such as OrderID 101, by summing the product quantities and unit prices from the OrderDetails table. Finally, we can apply discounts, track product price changes over time, and identify products with low stock levels by querying the Discount, ItemPriceHistory, and Product tables.

*** The 4th SELECT is not working due to an error I cannot seem to solve***

REFLECTION

During the process of creating the scripts, we faced several difficulties, particularly with getting the correct "grammar" and indentation in our SQL queries. It took an unexpected amount of time to ensure everything was properly formatted and structured, as even the smallest mistake in syntax could lead to errors in execution. We spent a lot of time troubleshooting issues like missing commas, incorrect joins, and misplaced parentheses, which were frustrating but necessary for the accuracy of the queries. Despite the challenges, the experience taught us the importance of precision in coding and the value of patience in debugging. This project was an incredibly challenging yet rewarding experience, we were thrilled with how it turned out! It was our first real attempt at converting a logical model into a physical one, and the process pushed us to think critically and creatively at every step. Despite the challenges we faced along the way, from crafting complex SQL queries to ensuring data integrity, the result exceeded our expectations. We learned so much about structuring databases, writing efficient queries, and solving real-world problems, which made the journey even more exciting. The sense of accomplishment we feel now makes all the challenging work worth it, and we are excited to apply these skills in future projects! We also genuinely appreciate each other for the teamwork and coordination that made taking down this project possible.