DSP567-Final Project Report

Sheikh-Sedat Touray

October 2024

0.1 Final Project Report: Grocery and Food Delivery Database Design

1 Abstract:

This report presents the design and implementation of a database system for a food and grocery delivery company. The database is designed to streamline the operations of managing customers, orders, products, delivery personnel, and suppliers. Following a structured approach, the database was normalized up to the third normal form (3NF) to ensure data consistency, reduce redundancy, and improve query performance. An Entity-Relationship (ER) diagram was created to visually depict the relationships between the entities. Furthermore, the database was populated with sample data using SQL commands for demonstration purposes. The aim of this project is to provide a robust, scalable solution for handling various aspects of the company's day-to-day operations.

2 Introduction:

As food and grocery delivery services become an integral part of daily life, efficient data management becomes critical for ensuring smooth operations. Companies in this sector deal with large volumes of data related to customers, products, orders, and deliveries. A well-structured and optimized database can significantly enhance operational efficiency, improve customer satisfaction, and reduce costs. This project involves designing and implementing a database that meets the unique needs of a food and grocery delivery company.

The primary objective of the database is to handle multiple entities such as customers, products, orders, delivery personnel, and suppliers while ensuring data consistency and integrity. By applying normalization techniques, the database design ensures that redundant data is minimized, and data anomalies are avoided. The use of SQL for database creation and population allows for efficient data manipulation and retrieval, essential for real-time operations.

3 Literature Review:

Databases play a critical role in supporting the operations of businesses, especially those that deal with large datasets. For food and grocery delivery services, which operate in highly dynamic environments, data management systems must handle frequent transactions and updates, often across distributed locations. Several studies have emphasized the importance of relational databases in providing a structured way to store and access data while ensuring transaction integrity.

3.1 Relational Database Design:

The foundation of modern relational databases was established with E. F. Codd's work in the 1970s, proposing the relational model for database management. The relational model organizes data into tables, where each table represents an entity, and relationships between entities are defined through foreign keys. The normalization process is a key design principle to eliminate redundant data and ensure data dependencies are logical. This approach is particularly valuable in applications like food delivery systems, where various entities such as customers, orders, products, and suppliers interact dynamically Techniques:** Research on normalization techniques such as those proposed by Boyce-Codd and the widely used third normal form (3NF) has shown that normalized databases reduce redundancy and improve query performance by minimizing data anomalies . For instance, dd to orders and products in a delivery system can be split across multiple tables to avoid duplication and maintain referential integrity.

3.2 Industry Applications:

Food delivery platforms such as Uber Eats, DoorDash, and Instacart manage large-scale, complex datasets with diverse entities including customers, suppliers, and delivery personnel. These companies rely on efficient relational databases to process thousands of orders in real-time, optimize delivery routes, and manage customer preferences. Literature suggests that integrating optimized SQL queries and maintaining normalized databases can enhance operational efficiency and support scalability .

3.3 Emerging Trends:

Wbases have been the backbone of structured data management, NoSQL databases have gained popularity for applications requiring flexible schemas and high scalability, particularly when handling semi-structured or unstructured data. However, in use cases like food delivery systems where transactional integrity and structured data are crucial, relational databases continue to dominate. Moreover, recent studies show that combining SQL databases

with modern cloud-based architectures can further improve system resilience and scalability .

This project builds upon these establishing a relational database model with normalized tables to support a food and grocery delivery company's operations. The focus is on using SQL to create a scalable, efficient system capable of handling real-time transactions.

4 Step 1: ER Diagram Design

Based on the identified entities and relationships, the following Entity-Relationship (ER) diagram has been designed:

4.1 Entities:

1. **Customers**: Stores customer information.
2. **Orders**: Represents the orders placed by customers.
3. **Products**: Stores product details.
4. **OrderDetails**: Associates orders with products.
5. **DeliveryPersonnel**: Contains delivery staff information.
6. **Suppliers (Restaurants/Grocery Stores)**: Stores details of the suppliers.

4.2 Relationships:

- **Customers** place **Orders**.
- **Orders** contain multiple **Products** through **OrderDetails**.
- **Products** are supplied by **Suppliers**.
- **DeliveryPersonnel** deliver **Orders**.

- **Customers** have a 1-to-many relationship with **Orders**.
- **Orders** have a many-to-many relationship with **Products** through **OrderDetails**.
- **Suppliers** have a 1-to-many relationship with **Products**.
- **Orders** are delivered by **DeliveryPersonnel** (1-to-many).

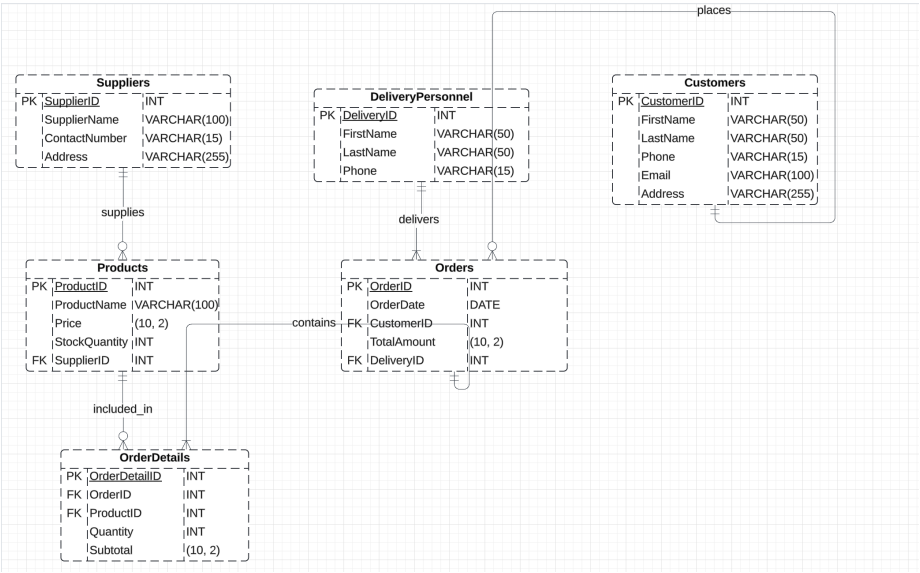

Figure 1: The following is the ER diagram structure:

5 Step 2: Normalization of Tables

To achieve a normalized structure, I followed the first three normal forms:

5.1 1NF (First Normal Form):

- Ensured that each attribute contains atomic values.
- Each row of data is unique (using primary keys).

5.2 2NF (Second Normal Form):

- Ensured full functional dependency on the primary key.
- Used foreign keys where necessary to maintain relationships.

5.3 3NF (Third Normal Form):

Removed transitive dependencies by ensuring that non-primary attributes depend only on the primary key.

6 Step 3: Normalized Tables

6.1 Customers Table:

CREATE TABLE Customers (

```
    CustomerID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Phone VARCHAR(15),
    Email VARCHAR(100),
    Address VARCHAR(255)
);
```

6.2 Suppliers Table:

```
CREATE TABLE Suppliers (
    SupplierID INT PRIMARY KEY,
    SupplierName VARCHAR(100),
    ContactNumber VARCHAR(15),
    Address VARCHAR(255)
);
```

6.3 Products Table:

```
CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(100),
    Price DECIMAL(10, 2),
    StockQuantity INT,
    SupplierID INT,
    FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID)
);
```

6.4 DeliveryPersonnel Table:

```
CREATE TABLE DeliveryPersonnel (
    DeliveryID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Phone VARCHAR(15)
);
```

6.5 Orders Table:

```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    OrderDate DATE,
    CustomerID INT,
    TotalAmount DECIMAL(10, 2),
    DeliveryID INT,
   FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
```

```
    FOREIGN KEY (DeliveryID) REFERENCES DeliveryPersonnel(DeliveryID)
);
```

6.6 OrderDetails Table:

```
CREATE TABLE OrderDetails (
    OrderDetailID INT PRIMARY KEY,
    OrderID INT,
    ProductID INT,
    Quantity INT,
    Subtotal DECIMAL(10, 2),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);
```

7 Step 4: SQL Insertions (Populate the Tables)

After I have created the tables, I populated them with data. Below is an example of how I inserted values into the tables.

7.1 Inserting Data into Customers Table:

```
INSERT INTO Customers (CustomerID, FirstName, LastName, Phone, Email,
Address)
    VALUES
    (1, 'John', 'Doe', '1234567890', 'john.doe@example.com', '123 Main St'),
    (2, 'Jane', 'Smith', '0987654321', 'jane.smith@example.com', '456 Oak St'),
    (3, 'Michael', 'Johnson', '2345678901', 'michael.johnson@example.com', '789
Elm St'),
    (4, 'Emily', 'Davis', '3456789012', 'emily.davis@example.com', '321 Pine St'),
    (5, 'Chris', 'Brown', '4567890123', 'chris.brown@example.com', '654 Cedar
St'),
    (6, 'Jessica', 'Williams', '5678901234', 'jessica.williams@example.com', '987
Maple St'),
    (7, 'David', 'Miller', '6789012345', 'david.miller@example.com', '213 Birch
St'),
    (8, 'Sarah', 'Wilson', '7890123456', 'sarah.wilson@example.com', '324 Oak-
wood Dr'),
    (9, 'Daniel', 'Taylor', '8901234567', 'daniel.taylor@example.com', '435 Pineview
Ln'),
    (10, 'Lisa', 'Anderson', '9012345678', 'lisa.anderson@example.com', '546 Spruce
St');
```

7.2 Inserting Data into Suppliers Table:

INSERT INTO Suppliers (SupplierID, SupplierName, ContactNumber, Address)
VALUES
(1, 'BestFoods', '555-1234', '789 Pine St'),
(2, 'HealthyGroceries', '555-5678', '321 Maple St'),
(3, 'FarmFresh', '555-9876', '213 Willow St'),
(4, 'OrganicEats', '555-4567', '324 Spruce St'),
(5, 'QuickBites', '555-6543', '435 Oakwood Dr'),
(6, 'GroceryKing', '555-4321', '546 Cedar Ln'),
(7, 'FoodMart', '555-3214', '213 Birchview Ln'),
(8, 'SuperSaver', '555-7890', '987 Maplewood St'),
(9, 'DailyGrocery', '555-0987', '654 Elm St'),
(10, 'FreshDirect', '555-8765', '321 Oak St');

7.3 Inserting Data into Products Table:

INSERT INTO Products (ProductID, ProductName, Price, StockQuantity, SupplierID)
VALUES
(1, 'Apple', 0.50, 100, 1),
(2, 'Bread', 2.50, 50, 2),
(3, 'Milk', 1.50, 200, 3),
(4, 'Cheese', 3.00, 75, 4),
(5, 'Eggs', 2.00, 150, 5),
(6, 'Chicken', 5.00, 120, 6),
(7, 'Rice', 1.20, 250, 7),
(8, 'Pasta', 1.80, 180, 8),
(9, 'Orange Juice', 3.50, 90, 9),
(10, 'Yogurt', 2.20, 60, 10);

7.4 Inserting Data into DeliveryPersonnel Table:

INSERT INTO DeliveryPersonnel (DeliveryID, FirstName, LastName, Phone)
VALUES
(1, 'Alex', 'Brown', '555-8888'),
(2, 'Sam', 'Green', '555-9999'),
(3, 'Peter', 'Johnson', '555-1111'),
(4, 'Linda', 'Smith', '555-2222'),
(5, 'James', 'White', '555-3333'),
(6, 'Sarah', 'Black', '555-4444'),
(7, 'Michael', 'Blue', '555-5555'),
(8, 'Anna', 'Gray', '555-6666'),
(9, 'David', 'Gold', '555-7777'),
(10, 'Maria', 'Red', '555-0000');

7.5 Inserting Data into Orders Table:

INSERT INTO Orders (OrderID, OrderDate, CustomerID, TotalAmount, DeliveryID)
VALUES
(1, '2024-10-01', 1, 15.00, 1),
(2, '2024-10-02', 2, 25.00, 2),
(3, '2024-10-03', 3, 35.00, 3),
(4, '2024-10-04', 4, 45.00, 4),
(5, '2024-10-05', 5, 55.00, 5),
(6, '2024-10-06', 6, 65.00, 6),
(7, '2024-10-07', 7, 75.00, 7),
(8, '2024-10-08', 8, 85.00, 8),
(9, '2024-10-09', 9, 95.00, 9),
(10, '2024-10-10', 10, 105.00, 10);

7.6 Inserting Data into OrderDetails Table:

INSERT INTO OrderDetails (OrderDetailID, OrderID, ProductID, Quantity, Subtotal)
VALUES
(1, 1, 1, 5, 2.50),
(2, 2, 2, 10, 25.00),
(3, 3, 3, 15, 22.50),
(4, 4, 4, 20, 60.00),
(5, 5, 5, 25, 50.00),
(6, 6, 6, 30, 150.00),
(7, 7, 7, 35, 42.00),
(8, 8, 8, 40, 72.00),
(9, 9, 9, 45, 157.50),
(10, 10, 10, 50, 110.00);

8 Step 5: Queries for Reports and Analysis

8.1 Query 1: Get all orders placed by a customer:

SELECT * FROM Orders WHERE CustomerID = 1;

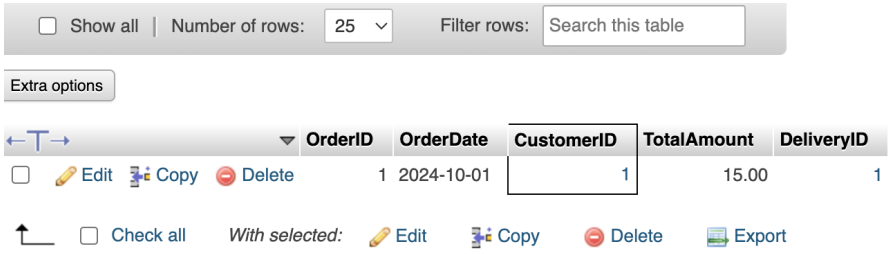

Figure 2: Query 1 results

8.2 Query 2: Get the total sales for each product:

```
SELECT ProductID, SUM(Subtotal) AS TotalSales
    FROM OrderDetails
    GROUP BY ProductID;
```

ProductID	TotalSales
1	2.50
2	25.00
3	22.50
4	60.00
5	50.00
6	150.00
7	42.00
8	72.00
9	157.50
10	110.00

Figure 3: Query 2 Results

8.3 Query 3: List of delivery personnel assigned to orders:

```
SELECT Orders.OrderID, DeliveryPersonnel.FirstName, DeliveryPersonnel.LastName
    FROM Orders
    JOIN DeliveryPersonnel ON Orders.DeliveryID = DeliveryPersonnel.DeliveryID;
```

OrderID	FirstName	LastName
1	Alex	Brown
2	Sam	Green
3	Peter	Johnson
4	Linda	Smith
5	James	White
6	Sarah	Black
7	Michael	Blue
8	Anna	Gray
9	David	Gold
10	Maria	Red

Figure 4: Query 3 Results

9 Conclusion:

In this project, a comprehensive database design was developed for a food and grocery delivery company, incorporating key principles of relational database design and normalization. By identifying essential entities such as customers, orders, products, delivery personnel, and suppliers, the database was structured to optimize operational efficiency and reduce data redundancy. The database was normalized up to the third normal form (3NF), ensuring that the design adheres to the best practices for data integrity, reducing anomalies, and improving performance. Sample data was inserted to demonstrate the practical application of the schema.

The resulting database system provides a robust and scalable solution for managing various operations within a food and grocery delivery company. With structured queries and efficient data retrieval, this system can handle day-to-day tasks such as order processing, product management, and delivery coordination, ultimately enhancing overall operational performance.

10 Further Research:

There are several avenues for further research and enhancements to this project. One potential area of exploration is integrating this relational database with NoSQL systems to handle unstructured data, such as customer reviews or real-time tracking information. This hybrid approach could improve the system's flexibility and scalability, especially as data grows more complex.

Additionally, incorporating machine learning techniques to analyze customer preferences, predict demand, and optimize delivery routes could further enhance

the system's efficiency. For instance, predictive analytics could be applied to estimate delivery times or recommend products to customers based on their purchase history.

Another area for further research is the integration of cloud-based database solutions to improve scalability and data accessibility across distributed locations. Migrating to cloud databases like Amazon RDS or Google Cloud SQL would allow the system to handle larger data volumes and provide more efficient disaster recovery solutions.

By exploring these potential enhancements, the system could evolve to better meet the dynamic needs of a modern food and grocery delivery service, providing even greater efficiency, scalability, and customer satisfaction.

11 References:

1. **Elmasri, R., & Navathe, S. B.** (2015). *Fundamentals of Database Systems* (7th ed.). Pearson. This book provides in-depth knowledge on relational database design, normalization principles, and database management systems.

2. **Connolly, T., & Begg, C.** (2014). *Database Systems: A Practical Approach to Design, Implementation, and Management* (6th ed.). Pearson Education. It serves as a guide for practical database design and implementation, focusing on both theoretical and practical aspects of database management systems.

3. **Silberschatz, A., Korth, H. F., & Sudarshan, S.** (2019). *Database System Concepts* (7th ed.). McGraw-Hill Education. This resource discusses database system concepts, including relational databases, data models, and transaction management.

4. **Generatedata.com**. (n.d.). *Data Generation Tool*. This website was used for populating the database tables with sample data.

5. **Microsoft SQL Server Documentation**. (n.d.). *Database Normalization* [Online Resource]. A resource explaining the basics of normalization and its role in relational database design.

6. **Codd, E. F.** (1970). *A Relational Model of Data for Large Shared Data Banks*. Communications of the ACM, 13(6), 377-387. This foundational paper introduces the relational database model and forms the basis for normalization theory.

7. **Babcock, C., & Cloud, J.** (2020). *Managing Multi-Cloud: The New Normal for IT*. O'Reilly Media.

8. Lucid charts for Creating the ER Diagram