**Database Management Systems Project Specification - Part II**

**Logical Schema Optimization and Unstructured Data Collection**

Fangru Li Chenxing Li Yuwen Zhong

[fl1099@nyu.edu](mailto:fl1099@nyu.edu) [cl7451@nyu.edu](mailto:cl7451@nyu.edu) [yz11737@nyu.edu](mailto:yz11737@nyu.edu)

**Introduction**

This project focuses on designing a robust database system for a food retailer website, structured into key modules: Customer, Order, Cart, and Product. These modules store detailed information to support essential operations like order management, product cataloging, and customer tracking.

To further enhance business strategies, a machine learning component predicts each user's membership level using tree-based models such as Random Forests or Gradient Boosted Trees. These models analyze user attributes, including purchase history and spending patterns, to categorize customers effectively.

The predicted membership levels enable the retailer to implement targeted marketing strategies, such as premium product recommendations for high-tier customers and loyalty incentives for lower-tier groups. This approach optimizes resource allocation, boosts customer satisfaction, and drives revenue growth, offering a scalable, data-driven solution for managing customer relationships.

**Database Schema Design Logic**

Considering that the database serves a food retail website, it should support the website's daily recording and querying functions. The database primarily includes four modules: Customer, Order, Product, and Cart. Each module contains corresponding relations that record more specific information. The following sections will elaborate on the design principles of each module and how they relate to other parts of the database.

* **Customer**

1. Account\_Info

The Account\_Info relation stores basic customer account information, including Customer\_id (PK), First\_Name, Last\_Name, Email\_address, Gender, Phone\_number, and Preferred\_Language. This schema acts as the primary repository for customer identity and contact details.

1. Customer\_Info

The Customer\_Info relation contains detailed customer membership and spending data, such as Membership\_id (PK), Customer\_id (FK), Avg\_Yearly\_Spend, Membership\_Status, Membership\_Level, and Points\_Accrued. This schema is designed to support customer loyalty programs and personalized services.

1. Customer\_Shipping\_Address

The Customer\_Shipping\_Address relation records shipping address details, including Customer\_id (FK), Shipping\_Address\_id (PK), Street, City, State, Country, Zipcode, and Phone\_Number. This schema ensures accurate and efficient delivery operations.

1. Customer\_Billing\_Address

Similarly, the Customer\_Billing\_Address relation holds billing address details, such as Customer\_id (FK), Billing\_Address\_id (PK), Street, City, State, Country, and Zipcode. It facilitates accurate billing and payment processing. Generally Speaking, one customer could have different shipping and billing addresses. Therefore, we record these different addresses with different relations.

1. Order\_History

The Order\_History relation captures customer order details, such as Order\_id (PK), Customer\_id (FK), Order\_Date, Total\_Amount, Item\_Amount, and Order\_Status. It tracks purchase history for analytics, order tracking, and recommendations.

1. Payment\_Method

The Payment\_Method relation stores customer payment details, including Customer\_id (FK), Payment\_id (PK), Billing\_Address\_id (FK), Card\_Type, Card\_Number, Expiry\_Date, and Cardholder\_Name. This schema supports secure and seamless payment processing.

* **Product**

1. Product SchemaThe Product schema contains the Product\_id, Name, SKU, detailed Description, Price, Category\_id, Supplier\_id, and Warehouse\_id. This schema serves as the central repository for basic product information and links to other schemas for category, supplier, and inventory management.
2. Category SchemaThe Category schema includes Category\_id, Category\_Name, and Description. It organizes products into different categories, allowing for easier classification and management of product groups.
3. Supplier SchemaThe Supplier schema comprises Supplier\_id, Supplier\_Name, Phone, Email, and Address. This schema provides details about the suppliers responsible for delivering products, enabling efficient supplier management.
4. Inventory SchemaThe Inventory schema stores Warehouse\_id, Product\_id, and Weight. It tracks the weight and storage of products within different warehouses, ensuring proper inventory control.
5. Discount SchemaThe Discount schema consists of Discount\_id, Product\_id, and Percentage. It defines the discount percentage for specific products, enabling dynamic pricing strategies.
6. Order\_Discount SchemaThe Order\_Discount schema contains Discount\_id, Order\_id, and Discount\_Amount. It links discounts to specific orders, tracking the total discount applied to each transaction.
7. Product\_Specialization SchemaThe Product\_Specialization schema includes Specialization\_id, Product\_id, Ingredient, Weight, and Expired. It provides detailed information about product variations, such as specific ingredients and expiration dates.
8. Product\_Review Schema

The Product\_Review schema contains Review\_id, Product\_id, Customer\_id, Rating, Comment, and Review\_date. It collects customer feedback, enabling insights into product performance and customer satisfaction.

* **Order**

1. **Cart Schema**

The Cart schema stores the CartID, Customer\_id, and CreatedDate, keeping track of each customer's shopping cart. It serves as the foundational structure to manage active shopping sessions and connects to the Cart\_items schema for detailed item listings within each cart.

1. **Cart Items Schema**

The Cart Items schema includes CartItemID, CartID, Product\_id, Quantity, Price, and Status. It records the specifics of products added to shopping carts, handling inventory directly from the cart level and allowing for dynamic updates to pricing and availability.

* **Cart**

1. **Order Schema**

The Order schema captures details with Order\_id, Product\_id, Shipping\_id, order\_date, receiver\_name, receiver\_email, and Customer\_id. It facilitates order processing and tracking, linking directly to customer and shipping information to ensure accurate delivery and order fulfillment.

1. **Order Feedback Schema**

The Order Feedback schema holds review\_id, Order\_id, Product\_id, rating, review\_date, review\_text, and recommend\_product. This schema collects customer reviews and ratings for purchased products, enhancing product credibility and providing valuable feedback for future customers.

1. **Shipping Information Schema**

The Shipping Information schema consists of shipping\_id, recipient\_name, Carrier, Shipping\_Address\_id, shipping\_date, and shipping\_status. It manages the logistics of product delivery, incorporating carrier information and detailed shipping statuses to streamline the distribution process.

1. **Return Refund Schema**

The Return Refund schema includes Order\_id, shipping\_id, return\_reason, refund\_amount, return\_date, and return\_status. It handles the complexities of product returns and refunds, documenting the reasons for returns and managing the financial implications associated with them.

**Data Lake Construction**

Since some information in the database (e.g., phone numbers, bank card information, and personal addresses) constitutes personal privacy, it is difficult to find corresponding content in a real public database. For the data collection part, all data in this database was generated using the data generation website Mockaroo (<https://www.mockaroo.com/>). When generating the data, we simulated real-world scenarios and appropriately added null values to some attributes.

**Logical Database Schema Optimization/Normalization**

1. Indexing: To improve query performance, we create indexes on Customer\_id , Order\_id , Product\_id and Shipping\_id. Indexing these columns will speed up search operations related to customer details, order history, product details and shipping status.
2. The database initially included separate tables for Cart\_save and Cart\_items, each handling different states of a customer's shopping process. These tables were consolidated into a single table called Cart\_items, with an additional column Status to indicate whether an item is currently active in the cart or saved for later. This not only reduced the complexity but also decreased the overhead associated with maintaining two similar data structures.
3. The Order table is bloated with fields like rating and return\_statue, which complicates updates and leads to data redundancy in the database. The Order table is structured into three key tables for managing different aspects of order data:
   1. **Order\_Base**: This table contains basic order details such as order\_id, customer\_id, and order\_date, serving as the central reference for each transaction.
   2. **Return\_Refund**: This table includes information related to returns and refunds, such as return\_reason and refund\_amount, providing insights into the reasons for returns and the financial impact on the business.
   3. **Order\_fb**: This table captures comprehensive feedback information with fields like Review\_id, Order\_id, Product\_id, Rating, Review\_date, Review\_text, and Recommend\_product. This structure helps the system gather detailed feedback on each product within an order, enhancing quality control and supporting customer satisfaction metrics.

**Unstructured Data**

The unstructured data in our project can be images of the products. To store images in MongoDB, each document in this collection should include the following fields:

* Product\_id: A unique identifier for the product, which will also serve as the link to the SQL database.
* Image\_URLs: An array to store the file paths or URLs of the product images.

The connection between MongoDB and SQL databases relies on the shared Product\_id field. First we need to ensure that the Product\_id in the SQL Product table matches the Product\_id in MongoDB’s product\_images collection. Then we can add a column in the SQL Product table, such as Has\_Images, to indicate whether a product has associated images stored in MongoDB.

To query product information and images, we can query the SQL Product table using the Product\_id to get structured data like name, price, and description, and we can query the MongoDB product\_images collection using the same Product\_id to fetch the associated image URLs or data.

Advantages of organizing unstructured data in MongoDB

* Data Separation: Structured data remains in the SQL database, while unstructured image data is stored in MongoDB, reducing the load on the SQL database.
* Flexibility: MongoDB’s schema-less design accommodates multiple images and additional metadata without altering the structure of the SQL database.
* Scalability: MongoDB is optimized for storing unstructured data, making it easy to manage a growing number of product images.