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

**Logical Schema Optimization and Machine Learning Model Creation**

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)

# **Project Background**

Modern enterprises increasingly rely on traditional relational databases to store structured data, while also incorporating real-time insights to drive business growth, achieve digital transformation, and enhance user experience and organizational efficiency. In the context of food retail, real-time responsiveness and personalized recommendations are critical for improving customer satisfaction and boosting order conversion rates.

The objectives of this phase include:

* Creating and optimizing the physical database model based on the logical database design developed in Part 2 for the food retail website.
* Developing a machine learning model to analyze shopping cart and order data, providing personalized recommendation solutions.
* Continuously refining the end-to-end reference architecture to support the seamless integration of business processes, applications, the DIKW (Data-Information-Knowledge-Wisdom) hierarchy, and infrastructure domains.

# **EDA Physical Database Design**

The design and optimization of a physical database are crucial for ensuring the performance, scalability, and efficiency of the food retail website. Building upon the logical schema from earlier phases, this stage involves translating the logical structure into a robust physical database implementation while applying various optimization techniques. To ensure the physical database design meets the performance, scalability, and efficiency requirements of the food retail website, several advanced optimization techniques are employed.

#### Indexing

Indexing plays a critical role in accelerating query performance by reducing the need for full table scans. Primary indexes are established on key fields like Customer\_ID, Product\_ID, and Order\_ID to enable rapid lookups and maintain data integrity. Secondary indexes target frequently queried attributes such as Order\_Date, facilitating efficient search operations. To optimize queries involving multiple fields, composite indexes are created for combinations like Customer\_ID and Order\_Status. These indexes collectively enhance the speed and efficiency of data retrieval, ensuring a responsive user experience even during peak usage.

#### Partitioning

Partitioning is employed to handle large datasets effectively by dividing tables into smaller, more manageable segments. For time-sensitive tables like Orders and OrderItems, range partitioning is applied based on OrderDate, allowing for faster historical data queries and simplified archival processes. List partitioning categorizes data based on attributes like Country or Product\_Category, enabling parallel processing for region-specific or product-category-specific queries. These partitioning strategies improve query performance and scalability while supporting efficient data maintenance.

#### Clustering

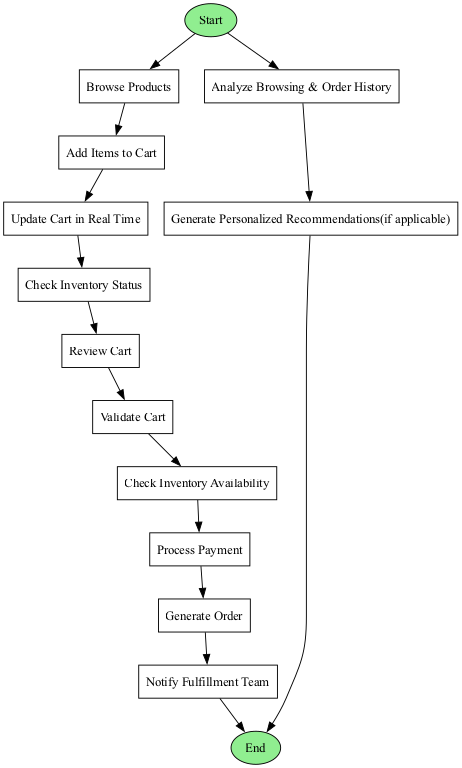
Clustering optimizes the physical storage of related data to minimize disk I/O operations during frequent joins and sequential data access. For example, Orders and Order\_Item tables are clustered around their shared Order\_ID, ensuring that data related to the same order is stored physically close. Similarly, the Cart and Cart\_Items tables are clustered to facilitate real-time updates and queries on shopping cart operations. This approach significantly reduces query execution time for operations that involve complex joins, enhancing overall database efficiency.

The integration of these optimization techniques—indexing, partitioning, clustering—ensures that the physical database design meets the operational and analytical needs of the food retail website. These enhancements enable faster query execution, efficient data storage, and seamless handling of large-scale transactional data, providing a robust foundation for real-time operations and strategic decision-making.

# **Business Use Cases and Workflow Design**

The database is designed to support critical business workflows essential for the operation of the food retail website, ensuring smooth customer interactions and efficient order management. Two primary use cases and one recommendation case are anticipated to guide the development of the system:

* Customer Browsing and Shopping Cart Management: Customers will be able to browse products and add items to their shopping cart. The system is expected to update the cart in real time, recalculating totals and reflecting the latest inventory status. This functionality is intended to provide an interactive and seamless shopping experience.
* Order Placement and Processing: Customers will review their shopping cart, confirm their order, and proceed to payment. The workflow will include validating the cart, checking inventory availability, processing payments, and generating the order. The system is designed to ensure data consistency and notify relevant departments for order fulfillment and delivery.
* Personalized Recommendations: Based on customer browsing history, cart contents, and previous orders, the system will generate tailored product recommendations. This functionality aims to enhance user engagement and drive sales through targeted suggestions.



These workflows are modeled using UML activity diagrams to visualize the sequence of actions, decision points, and system responses (see Figure). For example, the order placement workflow is expected to begin with a customer reviewing their cart, proceed through inventory checks and payment authorization, and conclude with order confirmation and notification.

The design of these use cases lays the foundation for a database system that supports key operational processes while enabling a responsive and user-friendly customer experience.

# **Machine Learning and Algorithm**

## Introduction and Use Case

The machine learning component of this project leverages customer behavior and transaction data to provide actionable insights through segmentation. By extracting and preprocessing data from key database tables, the system classifies customers into spending categories using a Decision Tree model. This segmentation supports business decisions by enabling targeted marketing campaigns, optimizing product recommendations, and prioritizing high-value customer groups.

For instance, in inventory management, the segmentation model identifies trending products among high-value customers, allowing the business to adjust inventory levels proactively and avoid stockouts or overstocking. Additionally, in dynamic pricing strategies, the model can highlight price sensitivity across different customer segments, helping the business tailor pricing models for maximum profitability.

These use cases, combined with efficient resource allocation, enhance customer retention through tailored engagement strategies and drive long-term customer satisfaction. By bridging raw data with strategic decision-making, this model serves as a critical tool for revenue growth, operational efficiency, and improved user experience.

## Algo Logic & implementation:

#### Data Preprocessing:

* + The data is extracted from multiple MySQL tables (customer\_info, account\_info, orders, and order\_item) using SQL JOINs to ensure all relevant fields are combined for analysis.
  + Missing or invalid data in critical fields like Avg\_Yearly\_Spend, Gender, and Price\_At\_Time\_Of\_Order are handled through numeric conversion and filling defaults.
  + The Price\_At\_Time\_Of\_Order is discretized into 5 quantile-based bins using pd.qcut to transform the continuous variable into ordinal categories (1 to 5), making it suitable for classification.

#### Model Training:

* + We used a DecisionTreeClassifier trained on the processed data, using Avg\_Yearly\_Spend and Gender as input features, and the quantile-based label as the target variable.
  + The trained model is serialized and saved as a .pkl file to allow for reuse without retraining on every request.

#### Prediction Logic:

* + The API endpoint /predict accepts user inputs via JSON requests.
  + Inputs are validated and encoded.
  + The trained model predicts the customer’s membership level, which is returned as a JSON response.

### Future Improvements:

1. Feature Expansion:
   * Integrate more complex features like customer demographics, policy details, and claim histories to improve the model’s predictive power.
2. Model Evolution:
   * Replace the Decision Tree with advanced models like Gradient Boosting Machines or Random Forests to improve accuracy and robustness.
   * Incorporate Deep Learning models for richer datasets.
3. Real-Time Predictions:
   * Deploy the model using streaming technologies like Apache Kafka or AWS Lambda to provide real-time predictions for incoming customer data.
4. Explainability:
   * Implement SHAP or LIME for feature importance analysis to make the model’s predictions transparent and interpretable.
5. Integration with Big Data Platforms:
   * Use platforms like Azure Synapse or Google BigQuery to scale and streamline the processing of massive datasets for enhanced insights.

# **Conclusion**

In conclusion, the combination of optimized physical database design and machine learning integration forms a robust framework for the food retail system. By leveraging advanced indexing, partitioning, and clustering techniques, the database ensures high performance, scalability, and efficiency. The decision tree-based machine learning model further enhances the system by enabling personalized recommendations and customer segmentation, driving both user satisfaction and revenue growth. With continuous refinement and the adoption of advanced technologies, this project lays the foundation for a responsive, intelligent, and future-ready food retail platform.