The Restaurant Chain Management System

Team Member:

GitHub:Github

Name	NUID
Hitesh Krishnappa	002471494
Michael Dereje	002542140
Yingying Zhuang	002304818
Yuxi Luo	002507445
Jingbo Ding	002308085

Updated ERD based on P2:

Based on professor's suggestion, we modified our ERD and here are main changes:

1.Delete entities Transaction, Revenue, and Sale Details.

2.Add new attributes "transaction amount","tip", "tax", and " table id" to the order entity.

3.Table is associated directly with the order.

4.remove Revenue and Finance entities.

5.For vendors, we distinguish between Vendor and Vendor Expense as two separate entities, where Vendor Expense is a subtype of Expense.

6.We've structured Ingredients inbound and outbound processes.

P3: The logical diagram

Based on the enhanced ERD diagram, we designed a logical diagram in third normal form. Here are the new design changes:

1. Removal of Redundant Financial and Transaction Entities

   Entities such as FINANCE, REVENUE, TRANSACTIONS, and SALE DETAIL are removed. Their functionalities, including transaction recording and revenue calculation, are integrated into relevant entities like ORDER and EXPENSE.

2. Atomic Attribute Optimization

   Like the PaymentInfo attribute in the Order entity is decomposed into atomic attributes including Transaction amount, tip, tax, and payment_status.

3. Expense and Vendor Relationships

   The EXPENSE entity is refined, the new entity called Vendor Expense is treated as a subtype of EXPENSE. The VENDOR entity is separated from transactional expense data, with Vendor Expense storing specific details and referencing VENDOR via the vendor_id foreign key.

4. Establishing One-to-Many Relationship in diagram

   For example, each ORDER is associated with one TABLE (via table_id as a foreign key in ORDER), and one TABLE can be linked to multiple ORDERS. It transforms a many-to-many relationship to a one-to-many relationship.

5. Ensuring Primary Keys, Foreign Keys, and draw diagram in 3NF

   All entities now have primary keys. Foreign keys are implemented for all relationships.

6. Resolved RESTAURANT and VENDOR Many-to-Many Relationship

   In the initial design, the relationship between RESTAURANT and VENDOR was unclear. We clarified that SUPPLIER_DELIVERY connects these two entities by documenting which vendor delivered to which restaurant. This design naturally forms the many-to-many relationship between vendors and restaurants through delivery transactions, allowing one vendor to deliver to multiple restaurants and one restaurant to receive from multiple vendors.

7. Added INVENTORY Entity

   The INVENTORY entity is a necessary component of a multi-restaurant system architecture. In a multi-restaurant environment, the same ingredient has different storage levels across different restaurants. For example, "eggs" may have 100 units in stock at the Boston restaurant while only 50 units at the New York restaurant. If storage quantities were stored directly in the INGREDIENT table, it would be impossible to distinguish inventory across different restaurants.

   Therefore, the INVENTORY entity uniquely identifies a specific ingredient's inventory at a specific restaurant through the combination of restaurant_id and ingredient_id, and maintains the real-time stock quantity (current_quantity) and last update timestamp (last_updated) for that combination.

8. Redefined Inbound and Outbound Process(implicit process)

   When a supplier delivery arrives, the system first records the delivery information (including restaurant ID) in the SUPPLIER DELIVERY table, then records the specific ingredients and quantities in the DELIVERY ITEM table. The system locates the corresponding inventory record in the INVENTORY table using the restaurant ID and ingredient ID, adds the delivered quantity to the current quantity, and updates the last_updated timestamp.

   When a customer order is placed, the system retrieves the ordered menu items through ORDER ITEM, then queries the recipe (required ingredients and quantities) for each menu item through MENU INGREDIENT. The system calculates the total requirement (quantity per serving × quantity ordered), then locates the corresponding inventory record in the INVENTORY table using the restaurant ID and ingredient ID, deducts the calculated requirement from the current quantity, and updates the last_updated timestamp.