Database Design Report

Food Manufacturing System (CSC 540)

Evie Kenna , Taylor Burke , Will Carter, Brett Hinkle

November 16, 2025

1 Executive Summary

This report documents the design and implementation of a comprehensive database system for a prepared/frozen meals manufacturing company. The system manages the complete lifecycle of food production, from ingredient sourcing through supplier relationships, recipe versioning, batch production, inventory tracking, and regulatory compliance.

2 Functional Dependencies That Drove the Design

We summarize the key functional dependencies (FDs) that shaped decompositions and keys. Primary/candidate keys are underlined.

Core Reference Entities

- **User**(user_id, first_name, last_name, role_code): user_id → first_name, last_name, role_code.
- **Category**(category_id, name, UNIQUE(name)): category_id ↔ name (unique alternative key).
- **Ingredient**(ingredient_id, ingredient_name, ingredient_type): ingredient_id → ingredient_name, ingredient_type.

Parties

- **Manufacturer**(manufacturer_id, manufacturer_name), with FK Manufacturer $\subseteq$ User: manufacturer_id → manufacturer_name.
- **Supplier**(supplier_id, supplier_name), with FK Supplier $\subseteq$ User: supplier_id → supplier_name.

Product & Structure

- **Product**(product_id, name, manufacturer_id, category_id, standard_batch_units): product_id → name, manufacturer_id, category_id, standard_batch_units.
- **IngredientComposition**(parent_ingredient_id, child_ingredient_id, quantity): (parent, child) → quantity, with parent $\neq$ child.

Recipes (Versioned BOM)

- **RecipePlan**(plan_id, product_id, manufacturer_id, version_no, creation_date, is_active; UNIQUE(product_id, manufacturer_id, version_no)): plan_id → product_id, manufacturer_id, version_no, creation_date, is_active.
  Note: Product enforces product_id → manufacturer_id, so RecipePlan contains a transitive FD; retained for uniqueness and fast joins.
- **RecipeIngredient**(plan_id, ingredient_id, quantity): (plan_id, ingredient_id) → quantity.

Lots, Batches, Consumption

- **ProductBatch**(batch_id, product_id, manufacturer_id, lot_number UNIQUE, quantity, unit_cost, production_date, expiration_date, plan_id): batch_id → all; lot_number is a candidate key generated by trigger. Product implies product_id → manufacturer_id (transitive dependency kept).
- **IngredientBatch**(batch_id, ingredient_id, supplier_id, lot_number UNIQUE, quantity_oz, on_hand_oz, unit_cost, expiration_date, intake_date): batch_id → all; lot_number candidate key.
- **BatchConsumption**(product_lot_number, ingredient_lot_number, quantity_consumed, consumption_date): (product_lot_number, ingredient_lot_number) → attributes.

Supplier Formulations (Versioned)

- **SupplierFormulation**(formulation_id, supplier_id, ingredient_id, version_no, pack_size, price_per_unit, effective_period_start_date, effective_period_end_date; UNIQUE(supplier_id, ingredient_id, version_no)): formulation_id → all.
- **SupplierFormulationMaterials**(formulation_id, ingredient_id, qty): (formulation_id, ingredient_id) → qty.

Business Rules

- **DoNotCombine**(ingredientA_id, ingredientB_id): key-only relation with symmetry captured by a check that stores an ordered pair (A < B).

3 Normalization Summary (BCNF vs 3NF)

All relations are at least 3NF; most satisfy BCNF. Two tables intentionally remain in 3NF to preserve useful foreign keys and reduce join costs while keeping redundancy controlled by FKs and procedures. Table located in Appendix C.

4 Constraints: Where Implemented and Why

4.1 Enforced in the DB (DDL, Checks, FKs, Triggers)

- **Keys and Referential Integrity:** PKs and FKs across entities (e.g., Product→Manufacturer; IngredientBatch→Supplier/Ingredient; RecipeIngredient→RecipePlan/Ingredient).

- **Domain/Business Checks:** Positive quantities/prices; supplier formulation date order; minimum shelf-life (expiration ≥ intake + 90 days); DoNotCombine stores ordered pair (A<B); uniqueness of natural keys (e.g., Category.name), lot numbers, and version triples.
- **Triggers:** Auto-generate lot numbers for ProductBatch/IngredientBatch; block consumption of expired ingredient lots; initialize on-hand quantity on receipt; decrement on-hand after consumption with underflow prevention.
- **Stored Procedures:** `record_production_batch` (atomic batch creation, cost roll-up, consumption posting); `trace_recall` (time-window trace from an ingredient lot to affected product lots); `evaluate_health_risk` (pairwise DoNotCombine check for an input ingredient set).

4.2 Implemented in Application Code (and Justification)

- **Role-Based Access and Menus:** Role is read from `User.role_code`, with routing to MANUFACTURER/SUPPLIER/VIEWER menus (Java). Sample queries do not require a User.role_code to view and are automatically routed to when selected. Authorization and UX are application concerns and change frequently; enforcing in code is appropriate.
- **Single Active Recipe Version per (product, manufacturer):** The application code ensures at most one recipe version is active (`is_active=TRUE`) at a time for each product. When a recipe version is set as active, all others are deactivated for that product (set newly active, flip others to FALSE in one transaction).

4.3 Could Not Be Fully Expressed in Bare Table Definitions

- **Transitive contamination/recall** requires joins and windowing ⇒ implemented in `trace_recall`.
- **Pairwise do-not-combine** across an arbitrary set ⇒ implemented in `evaluate_health_risk`.
- **Cost roll-up** for product batches ⇒ transactional procedure (`record_production_batch`).

5 Design Trade-offs and Rationale

- **BCNF vs 3NF Pragmatism:** Keeping `manufacturer_id` in **RecipePlan** and **ProductBatch** simplifies integrity checks/authorization and reduces joins; redundancy is bounded by FKs and procedures.
- **Versioning as First-Class:** Splitting plans/materials enables auditability and reproducibility (which recipe/supplier formulation version produced a batch).
- **Push Safety & Traceability Into the DB:** Triggers/procedures guarantee enforcement across all clients; UI/roles remain in the application.

Appendix A: ER Diagram

The final entity–relationship (ER) diagram for the Food Manufacturing System is shown below.

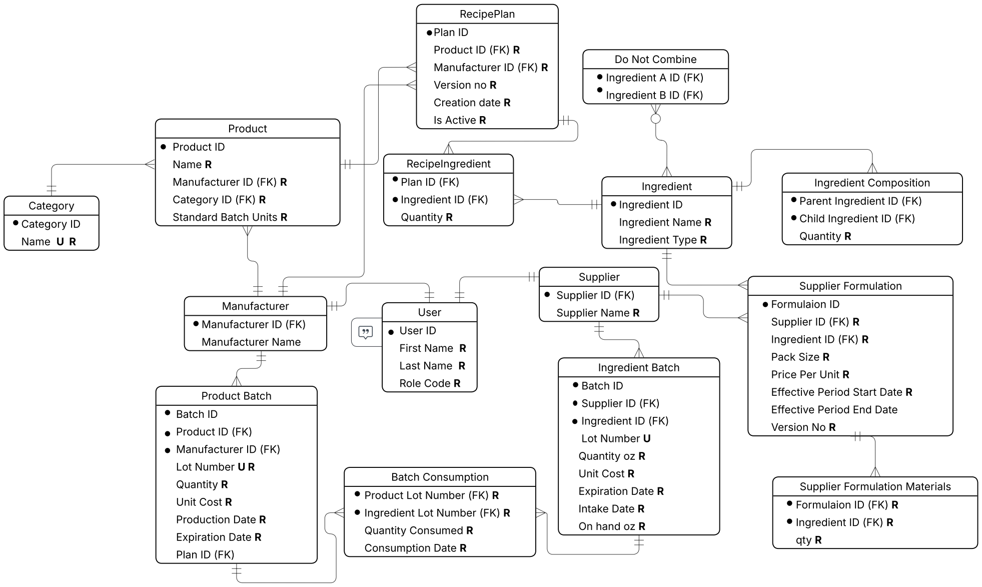

Figure 1: ER Diagram

Appendix B: Selected DDL / Triggers / Procedures

Listing 1: Example: IngredientComposition DDL

```sql
CREATE TABLE IngredientComposition (
  parent_ingredient_id INT,
  child_ingredient_id INT,
  quantity DECIMAL(10, 2) NOT NULL CHECK (quantity > 0),
  PRIMARY KEY (parent_ingredient_id, child_ingredient_id),
  CONSTRAINT parent_ingredient_id_fk FOREIGN KEY (parent_ingredient_id) REFERENCES
      Ingredient(ingredient_id),
  CONSTRAINT child_ingredient_id_fk FOREIGN KEY (child_ingredient_id) REFERENCES
      Ingredient(ingredient_id),
  CONSTRAINT parent_notequal_child CHECK (parent_ingredient_id != child_ingredient_id)
);
```

Listing 2: Example: Prevent expired consumption trigger

```sql
CREATE TRIGGER prevent_expired_consumption
BEFORE INSERT ON BatchConsumption
FOR EACH ROW
BEGIN
    DECLARE lot_expiration_date DATE;
    SELECT expiration_date INTO lot_expiration_date
    FROM IngredientBatch
    WHERE lot_number = NEW.ingredient_lot_number;
    IF NOW() > lot_expiration_date THEN
        SIGNAL SQLSTATE ’45000’
        SET MESSAGE_TEXT = ’You␣should␣not␣consume␣an␣expired␣ingredient␣lot.’;
    END IF;
END;
```

Appendix C: Normalization

Table	**Normal Form**	**Rationale**
User, Category, Ingredient, Manufacturer, Supplier	BCNF	PK fully determines non-keys; no transitive FDs. Category also enforces UNIQUE(name) (alt key).
IngredientComposition	BCNF	(parent, child) → quantity; no other FDs.
RecipePlan	**3NF**	plan_id → product_id, manufacturer_id and Product enforces product_id → manufacturer_id (transitive). Retained for uniqueness (product, manufacturer, version) and fast joins; could omit manufacturer_id to reach BCNF.
RecipeIngredient	BCNF	(plan_id, ingredient_id) → quantity.
ProductBatch	**3NF**	batch_id → product_id and manufacturer_id; Product implies product_id → manufacturer_id (transitive). Kept for clarity and FK checks; could drop manufacturer_id for BCNF.
IngredientBatch	BCNF	batch_id (and lot_number) determines all.
SupplierFormulation	BCNF	formulation_id (and composite alt key) determines all.
SupplierFormulationMaterials	BCNF	(formulation_id, ingredient_id) → qty.
BatchConsumption	BCNF	(product_lot_number, ingredient_lot_number) → attributes.
DoNotCombine	BCNF	Key-only relation; ordered pair prevents duplicates.

Table 1: Normalization summary. All tables are ≥3NF; most are BCNF.