Final Report: Grain Storage and Management Database System

Kerem KURU

Student ID: 220303016

Database Management Systems Project

Date: January 20, 2026

Abstract

This report presents the completed implementation of a *Grain Storage and Management Database System* designed to model real-world grain logistics in a multi-branch structure. The system manages regional branches, silos and compartments, farmer/plot/harvest records, supplier deliveries (intakes), lot creation, quality testing, stock movements, and customer sales with lot-based allocations. The database is implemented in Microsoft SQL Server with a normalized relational schema, strong referential integrity, unique business constraints, and triggers that enforce critical rules such as capacity limits, non-negative occupancy, allocation constraints, and automated delivery movements when orders are closed. Sample data and test scenarios confirm consistent traceability from intake/lot to storage, movement, and final customer delivery.

Arel Üniversitesi, Müh. Mim. Fak. Bilgisayar Mühendisliği.
Türkoba Mah., Erguvan Sok., No:26, K 34537, Tepekent – Büyükçekmece, İstanbul.

1 Introduction

Grain logistics requires precise tracking of *origin*, *quality*, *storage conditions*, and *inventory movements* to prevent losses and ensure reliable supply. In large-scale operations, the same product may be distributed across multiple branches and storage compartments, while orders must be fulfilled without breaking traceability rules. To address these needs, this project implements a complete relational database called **GrainStorageDB**.

The system models a multi-branch structure with silos and compartments, integrates farmer and harvest information, records intakes at specific branches, creates lots linked 1:1 with intakes, stores laboratory quality test results, and tracks all stock changes through movements. On the sales side, customer orders are recorded using order headers and lines, and sales are fulfilled by allocating specific lots. The implementation focuses on strong data integrity, normalization, and enforcement of business rules using foreign keys, unique constraints, computed values, and triggers.

2 Procedure

1. **Requirements & workflow definition:** Defined the operational steps from harvest/intake to storage, movement, and sales delivery, including traceability and capacity constraints.
2. **Conceptual modeling (EER):** Identified core entities (Branch, Silo, Compartment, Product, Supplier, Farmer, Plot, Harvest, Intake, Lot, QualityTest, Movement, Customer, SalesOrder, SalesOrderLine, Allocation) and their cardinalities.
3. **Logical design & normalization:** Converted the conceptual design into a relational schema with 3NF structure; separated headers/lines and event tables.
4. **Implementation in MS SQL Server:** Created tables, primary keys, foreign keys, and unique business rules (e.g., unique allocation per order line+lot).
5. **Business rule enforcement via triggers:** Enforced capacity, non-negative occupancy, lot outbound limits, allocation limits, and automatic delivery actions when orders close.
6. **Data population & testing:** Inserted sample data and validated correctness with scenario tests and integrity checks.

Technology Used: Microsoft SQL Server (T-SQL), EER diagram tool (draw.io), sample dataset scripts.

3 Theory

The database is designed according to relational database principles and normalized to Third Normal Form (3NF). The overall structure follows an event-based modeling approach to ensure full traceability of grain flows from intake to final delivery.

Core **master entities** represent relatively stable objects in the system, including `BRANCH`, `SILO`, `SILO_COMPARTMENT`, `PRODUCT`, `SUPPLIER`, `CUSTOMER`, `FARMER`, and `FARM_PLOT`. These entities define the physical storage infrastructure, actors, and reference data required for grain logistics.

Operational event entities capture all time-dependent activities in the system. These include `HARVEST`, `INTAKE`, `LOT`, `QUALITYTEST`, `MOVEMENT`, `SALESORDER`, `SALESORDERLINE`, and `ALLOCATION`. Each intake generates exactly one lot (1:1 relationship), ensuring strict traceability. Quality tests are linked to lots and may occur multiple times during storage.

Inventory changes are handled exclusively through the `MOVEMENT` entity, which records inbound, transfer, and outbound operations. This design guarantees that silo and compartment occupancies are always derived from movement data rather than being updated directly.

To enforce business rules that go beyond standard constraints, the system uses database triggers. These triggers maintain branch-level capacity totals when silo data changes, prevent negative stock and over-capacity situations during movements, ensure that allocations do not exceed ordered or available quantities, and automatically generate allocation and delivery movements when a sales order is closed and sufficient stock exists.

4 Results

Testing with the sample dataset confirmed that the database preserves referential integrity and supports end-to-end traceability. Representative outcomes:

- Branch capacities are consistent and updated when silo records change.
- Silo occupancy always remains within $[0, \text{capacity}]$ due to movement validation.
- A lot can be tracked from intake through quality tests and compartment movements to final delivery.
- Closing an order triggers automated allocation and outbound movement (or blocks closure if stock is insufficient).

Table 1: Example of Branch Silo Capacities (from the implemented dataset)

Branch Code	**City / Region**	**Silo Capacity (tons)**
IST-CTL	İstanbul / Marmara	20,000
KNY-MRK	Konya / İç Anadolu	22,000
ADA-MRK	Adana / Akdeniz	18,000
GDA-DIY	Diyarbakır / Güneydoğu Anadolu	22,000

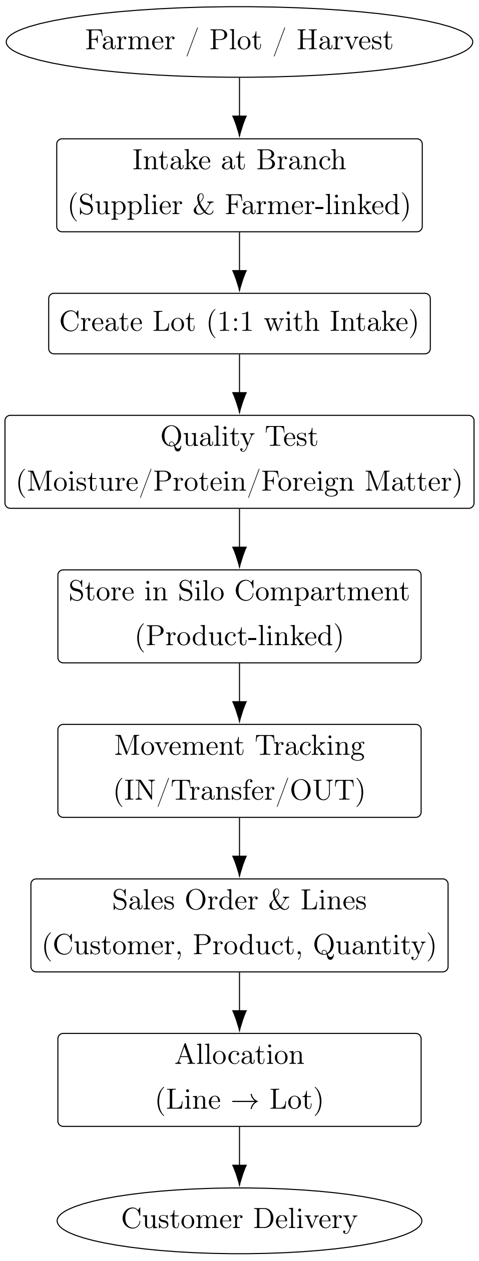
Farmer / Plot / Harvest
Intake at Branch (Supplier & Farmer-linked)
Create Lot (1:1 with Intake)
Quality Test (Moisture/Protein/Foreign Matter)
Store in Silo Compartment (Product-linked)
Movement Tracking (IN/Transfer/OUT)
Sales Order & Lines (Customer, Product, Quantity)
Allocation (Line → Lot)
Customer Delivery

Figure 1: End-to-end process flow implemented in the database

5 Remarks and Recommendations

The completed system provides a consistent and scalable foundation for grain logistics with strong integrity guarantees. Suggested future improvements:

- **IoT integration:** Store compartment sensor readings (temperature/humidity/CO_2) and link them to lots.
- **Analytics layer:** Predict spoilage risk using quality tests + storage duration + environmental data.
- **Security & auditing:** Add users/roles and a change-log table for critical operations.
- **Performance scaling:** Add indexes/partitioning strategies for large movement and order volumes.

References

1. Elmasri, R., Navathe, S. B. *Fundamentals of Database Systems*, 7th Ed., Pearson, 2016.
2. Connolly, T., Begg, C. *Database Systems: A Practical Approach*, Pearson, 2015.
3. Microsoft. *SQL Server Documentation: Constraints, Indexes, and Triggers*. `https://learn.microsoft.com/en-us/sql/`
4. GeeksforGeeks. "Introduction to EER Diagram Model in DBMS." `https://www.geeksforgeeks.org/dbms/introduction-of-er-model/`
5. dbdiagram.io. "Online EER Diagram & Database Design Tool." `https://dbdiagram.io/`