Inventory Management System

Team Name: **Schema Squad**

Nikhil Gupta
Computer Science and Engineering
University at Buffalo
Buffalo, USA
ngupta22@buffalo.edu

Rahul Karkala Kudva
Computer Science and Engineering
University at Buffalo
Buffalo, USA
rahulkar@buffalo.edu

Indushree Byrareddy
Computer Science and Engineering
University at Buffalo
Buffalo, USA
indushre@buffalo.edu

***Abstract*—Businesses need effective inventory management to satisfy consumer needs and stay competitive. Conventional techniques that rely on Excel files or manual processes can result in errors and inefficiency. This project suggests using a relational database to create an Inventory Management System (IMS) to address these issues. With real-time information to optimize inventory levels and lower operating costs, the IMS seeks to simplify order processing, warehouse management, and inventory tracking. The IMS uses database technology to provide scalability, data consistency, and sophisticated query capabilities that help firms increase overall operational efficiency, reduce stockouts, and improve inventory accuracy. The significance of this project is found in its ability to transform inventory management procedures throughout industries by providing companies with a centralized platform to efficiently manage inventory and promote long-term growth.**

I. Problem Statement

Managing inventory efficiently is crucial for businesses to maintain optimal operations and meet customer demands. However, traditional methods of inventory management using Excel files often lead to inefficiencies, errors, and lack of scalability. Our project aims to address these challenges by developing a comprehensive inventory management system using a relational database.

Why not Excel? Excel files have limitations in handling large volumes of data and concurrent users. A database provides scalability to accommodate growing data and user needs. Databases enforce data integrity constraints, ensuring accurate and consistent data storage and retrieval. Databases offer powerful query capabilities for complex data analysis and reporting compared to Excel's basic functionalities. Databases support concurrent access by multiple users while maintaining data consistency, which is challenging with Excel files.

Background and Objectives: The inefficiencies associated with manual inventory management can lead to stockouts, overstocking, and increased operational costs. Our objective is to develop a robust inventory management system that streamlines inventory tracking, order processing, and warehouse management. By automating these processes and providing real-time insights, our system aims to improve inventory accuracy, reduce stockouts, minimize holding costs, and enhance overall operational efficiency.

Contribution to the Problem Domain: Our project's contribution lies in its potential to transform inventory management practices across industries. By leveraging database technology, we can provide businesses with a centralized platform to manage inventory effectively, leading to cost savings, improved customer satisfaction, and sustainable growth. Inventory management is essential to supply chain optimization and corporate competitiveness, which highlights the importance of this contribution.

II. Target User

In a retail company, warehouse managers use the inventory management system to oversee multiple warehouses' operations, track inventory levels, and ensure timely order fulfillment. Sales representatives access the system to process customer orders, check product availability, and provide accurate delivery estimates. Supply chain managers leverage the database to analyze sales data, forecast demand, and coordinate with suppliers to maintain optimal inventory levels. Database administrators manage user accounts, perform routine maintenance tasks, and ensure the database's smooth operation. The target users for the inventory management system database include:

A. Warehouse Managers

Warehouse managers will use the database to monitor inventory levels, track stock movements, and manage warehouse operations efficiently. They will be responsible for updating inventory records, generating reports, and optimizing warehouse layout based on inventory data.

B. Sales and Customer Service Representatives

Sales and customer service representatives will utilize the database to process orders, check product availability, and provide accurate information to customers regarding order status and delivery schedules.

C. Supply Chain Managers

Supply chain managers will rely on the database to analyze inventory trends, forecast demand, and make informed decisions regarding procurement, production planning, and inventory replenishment strategies.

D. Administrators

Database administrators will administer the database, ensuring its security, performance, and integrity. They will be responsible for tasks such as user access management, backup and recovery, and database optimization.

III. E/R Diagram

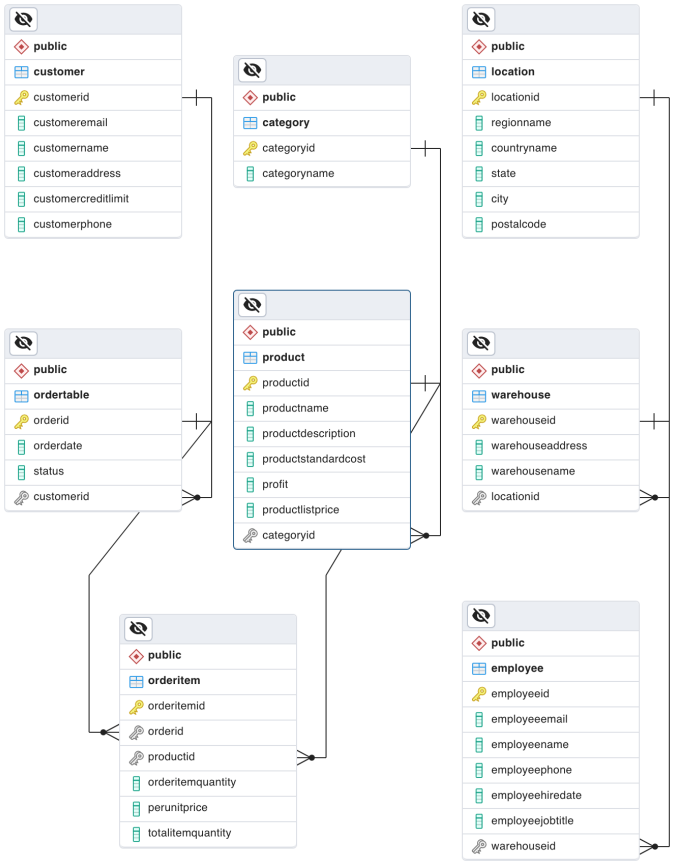

Fig. 1. E/R Diagram

IV. Database Table Structure

A. Location Table

- **LocationID** (Primary Key, INT): Unique identifier for each location.
- **RegionName** (VARCHAR(255)): Name of the region.
- **CountryName** (VARCHAR(255)): Name of the country.
- **State** (VARCHAR(255)): State name.
- **City** (VARCHAR(255)): City name.
- **PostalCode** (VARCHAR(255)): Postal code of the location.

B. Warehouse Table

- **WarehouseID** (Primary Key, INT): Unique identifier for each warehouse.
- **WarehouseAddress** (VARCHAR(255)): Address of the warehouse.
- **WarehouseName** (VARCHAR(255)): Name of the warehouse.
- **LocationID** (Foreign Key, INT): References the Location table's LocationID.
- *Actions on Foreign Key*: No action on delete.

C. Employee Table

- **EmployeeID** (Primary Key, INT): Unique identifier for each employee.
- **EmployeeEmail** (VARCHAR(255)): Email address of the employee.
- **EmployeeName** (VARCHAR(255)): Name of the employee.
- **EmployeePhone** (VARCHAR(255)): Phone number of the employee.
- **EmployeeHireDate** (DATE): Date when the employee was hired.
- **EmployeeJobTitle** (VARCHAR(255)): Job title of the employee.
- **WarehouseID** (Foreign Key, INT): References the Warehouse table's WarehouseID.
- *Actions on Foreign Key*: No action on delete.

D. Category Table

- **CategoryID** (Primary Key, INT): Unique identifier for each category.
- **CategoryName** (VARCHAR(255)): Name of the category.

E. Product Table

- **ProductID** (Primary Key, INT): Unique identifier for each product.
- **ProductName** (VARCHAR(255)): Name of the product.
- **ProductDescription** (TEXT): Description of the product.
- **ProductStandardCost** (DECIMAL(10,2)): Standard cost of the product.
- **Profit** (DECIMAL(10,2)): Profit margin of the product.
- **ProductListPrice** (DECIMAL(10,2)): List price of the product.
- **CategoryID** (Foreign Key, INT): References the Category table's CategoryID.
- *Actions on Foreign Key*: No action on delete.

F. Customer Table

- **CustomerID** (Primary Key, INT): Unique identifier for each customer.
- **CustomerEmail** (VARCHAR(255)): Email address of the customer.
- **CustomerName** (VARCHAR(255)): Name of the customer.
- **CustomerAddress** (TEXT): Address of the customer.

- **CustomerCreditLimit** (DECIMAL(10,2)): Credit limit of the customer.
- **CustomerPhone** (VARCHAR(255)): Phone number of the customer.

G. OrderTable Table

- **OrderID** (Primary Key, INT): Unique identifier for each order.
- **OrderDate** (DATE): Date when the order was placed.
- **Status** (VARCHAR(255)): Status of the order.
- **CustomerID** (Foreign Key, INT): References the Customer table's CustomerID.
- *Actions on Foreign Key*: No action on delete.

H. OrderItem Table

- **OrderItemID** (Primary Key, INT): Unique identifier for each order item.
- **OrderID** (Foreign Key, INT): References the OrderTable table's OrderID.
- **ProductID** (Foreign Key, INT): References the Product table's ProductID.
- **OrderItemQuantity** (INT): Quantity of the ordered item.
- **PerUnitPrice** (DECIMAL(10,2)): Price per unit of the ordered item.
- **TotalItemQuantity** (INT): Total quantity of the ordered item.
- *Actions on Foreign Keys*: No action on delete.

V. BCNF Check

To ensure that all relations are in Boyce-Codd Normal Form (BCNF), we need to identify functional dependencies and eliminate any dependencies on non-superkey attributes. Below are the dependencies for each relation:

A. Location

- Dependencies:
  - $LocationID \rightarrow RegionName$, $CountryName, State$, $City, PostalCode$
- All attributes are functionally dependent on the $LocationID$ (primary key).
- No partial dependencies or transitive dependencies exist.

B. Warehouse

- Dependencies:
  - $WarehouseID \rightarrow WarehouseAddress$, $WarehouseName, LocationID$
  - $LocationID \rightarrow RegionName$, $CountryName, State$, $City, PostalCode$
- The $WarehouseID$ uniquely determines $WarehouseAddress$ and $WarehouseName$.
- $LocationID$ determines $RegionName, CountryName, State, City$, $PostalCode$.
- No partial dependencies or transitive dependencies exist.

C. Employee

- Dependencies:
  - $EmployeeID \rightarrow EmployeeEmail$, $EmployeeName, EmployeePhone$, $EmployeeHireDate, EmployeeJobTitle$, $WarehouseID$
- The $EmployeeID$ uniquely determines all other attributes.
- No partial dependencies or transitive dependencies exist.

D. Category

- Dependencies:
  - $CategoryID \rightarrow CategoryName$
- The $CategoryID$ uniquely determines $CategoryName$.
- No partial dependencies or transitive dependencies exist.

E. Product

- Dependencies:
  - $ProductID \rightarrow ProductName$, $ProductDescription, ProductStandardCost$, $Profit, ProductListPrice$, $CategoryID$
  - $CategoryID \rightarrow CategoryName$
- $ProductID$ uniquely determines all other attributes.
- $CategoryID$ determines $CategoryName$.
- No partial dependencies or transitive dependencies exist.

F. Customer

- Dependencies:
  - $CustomerID \rightarrow CustomerEmail$, $CustomerName, CustomerAddress$, $CustomerCreditLimit, CustomerPhone$
- The $CustomerID$ uniquely determines all other attributes.
- No partial dependencies or transitive dependencies exist.

G. OrderTable

- Dependencies:
  - $OrderID \rightarrow OrderDate$, $Status, CustomerID$
- The $OrderID$ uniquely determines $OrderDate$, $Status$, and $CustomerID$.
- No partial dependencies or transitive dependencies exist.

H. OrderItem

- Dependencies:
  - $OrderItemID \rightarrow OrderID$, $ProductID, OrderItemQuantity$, $PerUnitPrice, TotalItemQuantity$
- The $OrderItemID$ uniquely determines all other attributes.
- No partial dependencies or transitive dependencies exist.

Conclusion: All tables appear to be in BCNF as there are no non-trivial functional dependencies on any proper subset of the primary keys. Therefore, no decomposition is necessary.

REFERENCES

[1] https://www.kaggle.com/datasets/hetulparmar/inventory-management-dataset
[2] https://www.postgresql.org/docs/current/index.html
[3] https://www.pgadmin.org/docs/pgadmin4/8.3/index.html
[4] https://www.pgadmin.org/docs/pgadmin4/latest/erd_tool.html