Members: Quoc Nhan Tra, Binod Chhantyal, Solomon Rosenfeld
Date: 10/31/2025
Course: CSCE 4350.001

Project Phase 1: Project Proposal

Project Description

The Retail Auto Parts System project is a database centric application designed by undergrad students to computerize the operations of a small auto parts company. The hypothetical premise of this project is that the business is currently operating with paper records using limited computational capabilities. Our system will be designed to process:

- Customer orders and payments. (In Customer Mode)
- Record inventory management, purchases, and supplier records. (In Store Mode)
- Employee management and reports. (In Store Mode)

Purpose of phase I

The purpose of this phase is to identify and document all problems, solutions, hardware choices, and software choices. It will also document the project timeline, the System Requirement Analysis, and all team member's roles and contributions. This proposal will articulate the concept of the Retail Auto Parts System into a clear actionable plan with a timeline. By evaluating and establishing all aspects of the system into a clear plan, the team will have a structured method for tracking progress.

Project Timeline

Project Phase	Tasks	Due Date	Deliverables
Phase 1	Analysis and Conceptual Design	10/03/2025	Proposal report
Phase 2	Logical and Physical database design	10/31/2025	ER diagram, Schema Diagram
Phase 3	System implementation and testing	11/21/2025	Final System Product
Phase 4	Final Presentation	12/05/2025	Power Point Presentation and Demonstration

Problem Encountered and Solutions

Problem 1: Technology Stack Selection
Choosing between different DBMS options (MySQL, PostgreSQL, MongoDB) and web frameworks
Solutions: After evaluating requirements, we selected PostgreSQL for its robust ACID compliance, excellent handling of complex queries, and strong support for relational data modeling essential for inventory management. For the web framework, we chose a modern stack that balances performance with developer productivity.
Why choose PostgreSQL: PostgreSQL offers some advanced features like JSONB support for flexible data storage, transaction handling, and scalability. Additionally, it's open source, well-documented, and has strong community support.

Hardware Requirements

Workstation: Laptop or Desktop
Operating System: Windows/macOS/Linux
Network: connectivity between frontend, backend, and database.

Software and Technology Stack

Database Management System

- PostgreSQL
- Enterprise-grade features, ACID compliance, excellent performance with complex queries
- pgAdmin 4 for database administration

Backend Development

- Framework: Python with Django
- RESTful API design

Frontend Development

- HTML/CSS/Javascript
- Django templates for web interface, creates menu system

System Requirement Analysis

The system must have two modes of operations, “Customer mode” and “Store mode”.
Customer Mode:

- The system must allow ordering products online
- The system shall support payment
- The system shall store customer’s personal information (name, address, phone, email) for billing.
- The system shall have user and password authentication (for both customers and employees).
- The system shall allow customers to browse existing products.
- The system shall contain a search system.
- The system shall display details when a product is selected (price, quantity, new/used, ect).
- The system shall store customer history activity.
- The system shall have a “shopping cart” function.

Store mode:

- The system shall store employee information (add/change/delete).
- The system shall have employee log in authentication.
- The system must have a section to:
    - Order new products, reorder, cancel orders, and handle returns.
    - Update database for personnel and parts information.
    - Generate reports on products and employee status.
    - Include delivery information such as order date, delivery date, payment method, and cancellations.

The system shall be menu-driven.

Database Design Overview

Initial draft identifying core entities:

- Customer(Customer_ID, Customer_name, Address, Phone, Email, Username, Password)
- Employee(Employee_ID, Employee_name, Role, Login_ID, Password)
- Auto Parts(Part_ID, Part_name, Quantity, Category, Condition)
- Order(Oder_ID, Customer_ID, Order_date, Payment_Method, Total_amount)
- Supplier(Delivery, Supplier_name, Supplier_Contact_info)
- Delivery(Delivery_ID, Order_ID, Date, Status.)

Phase I: Planning and Analysis

- Requirements gathering
- Technology stack selection
- Team formation and role assignment
- Initial documentation
- Deliverable: Phase I Report

Phase II: Database Design and Implementation

- Create detailed Entity-Relationship (ER) diagrams
- Design normalized relational schema with all constraints
- Implement database in PostgreSQL
- Populate with sample/test data
- Validate database design through testing

Phase III: System Development, Testing & Deployment

- Develop backend API and business logic
- Create frontend user interface
- Integrate all system components
- Perform comprehensive testing
- Deploy to production environment
- Create user documentation

Phase IV: Final Presentation

- Present the complete project to the class
- Demonstrate functionality and project outcomes

Team Member Contribution

Quoc Nhan Tra – Requirements & Frontend Specialist

- Phase I (Analysis & Conceptual Design):
    - Collect user requirements (customer/store mode features).
    - Draft system requirement analysis section.
    - Help with hardware/software/DBMS selection.
- Phase II (Database Design):
    - Contribute to ER diagrams by focusing on customer- and store-facing entities (e.g., Customers, Orders, Payments).
    - Help normalize relations and refine schema.

- Phase III (Implementation & Testing):
    - Develop frontend (UI/UX) for customer/store menus.
    - Connect frontend forms to backend API.
    - Write test cases for user interaction (login, search, ordering).
- Programming Focus: Frontend development, user input validation, integration with backend.

Binod Chhantyal – Database & Backend Specialist

- Phase I:
    - Research DBMS options and justify choice (MySQL, PostgreSQL, etc.).
    - Contribute to system plan and data flow.
- Phase II:
    - Contribute to ER diagram and schema design (tables, keys, constraints).
    - Insert sample data for testing.
- Phase III:
    - Implement backend API (CRUD, authentication, reports).
    - Optimize SQL queries and indexing.
    - Test queries and backend response.
- Programming Focus: Database schema creation, SQL queries, backend API.

Solomon Rosenfeld – System Integrator & Tester

- Phase I:
    - Document project plan and timeline.
    - Draft Phase I proposal with contribution requirements with other members.
- Phase II:
    - Assist with schema designs and integrity checks (foreign keys, normalization).
    - Prepare ER/schema diagrams for the report.
- Phase III:
    - Develop/Program business logic (order processing, employee management).
    - Develop system integration (connecting frontend, backend, and database).
    - Lead system testing & demonstrations
- Programming Focus: Middleware code, integration testing, error handling.

Project Phase 2: Logical and Physical Database Design

Introduction/Purpose of Phase II

The purpose of this phase is to expand on the problems encountered in Phase I, document revisions to the Phase I report, and team member contribution. It will also document the ER Diagrams, Schema Diagram, and the results of the database schema creation. By expanding on these aspects, the team will be able to further refine the architecture of the software and improve the problematic aspects.

Phase 1 and Revisions

Initially, we agreed on using PostgreSQL as our DBMS. However, we encountered problems while using pgAdmin 4, and in the end decided to switch to DBeaver as our database administration tool, as it is more intuitive and functional. Additionally, after analyzing the project requirements, we identified additional core entities and attributes which were added to the ER diagram and relational schema.

Problems Encountered and Solutions

Problem 1: Relationship complexities, such as some many to many relationships caused anomalies.
Solution: In order to clear this problem, we created junction tables such as the case with Inventory and Orderitem with composite primary keys. For example, the primary key in Inventory is created with (store_id, part_id), which becomes a unique key. This method helps eliminate duplicates and helps maintain integrity.

Problem 2: When we first started defining objects we faced difficulty determining which should become entities and which should be attributes. Some data overlapped, and several attributes were unintentionally excluded. Each time we revised the ER diagram, new dependencies appeared that we forgot which requires updating the schema.
Solution: We had to go back and carefully review the project requirements and clearly identify the entity's purpose. In the end we created a list defining the entities and attributes one by one as required by the project outline, then began to proceed from there.

Documentation Produced in Phase II

Defining entities and attributes

- **Store** — store_id (PK), name, phone, address_line1/2, city, state, postal_code
- **Customer** — customer_id (PK), full_name, customer_email, customer_phone, created_at, username, password
- **Employee** — employee_id (PK), store_id (FK), full_name, role, email, username, password
- **Supplier** — supplier_id (PK), name, contact_email, phone, Address
- **Auto_Part** — part_id (PK), name, category, condition, unit_price, reorder_level
- **Inventory** — (store_id, part_id) (PK, FK), quantity_on_hand
- **Purchase_Order**— po_id (PK), store_id (FK), supplier_id (FK), order_date, expected_date, status
- **Order_Item** — (po_id, part_id) (PK, FKs), quantity, unit_cost
- **Customer_Order** — order_id (PK), customer_id (FK), store_id (FK), order_date, status
- **OrderItem** (assoc.) — (order_id, part_id) (PK, FKs), qty, unit_price
- **Payment** — payment_id (PK), order_id (FK), payment_method, amount, paid_date, card_last_four_digit, authentication_code
- **Delivery** — delivery_id (PK), order_id (FK), ship_date, delivery_date, employee_id(FK), tracking_number, deilvery_status
- **ReturnItem** — return_id (PK), order_id (FK), part_id (FK), quantity, reason, created_date

Entity-Relationship Diagram

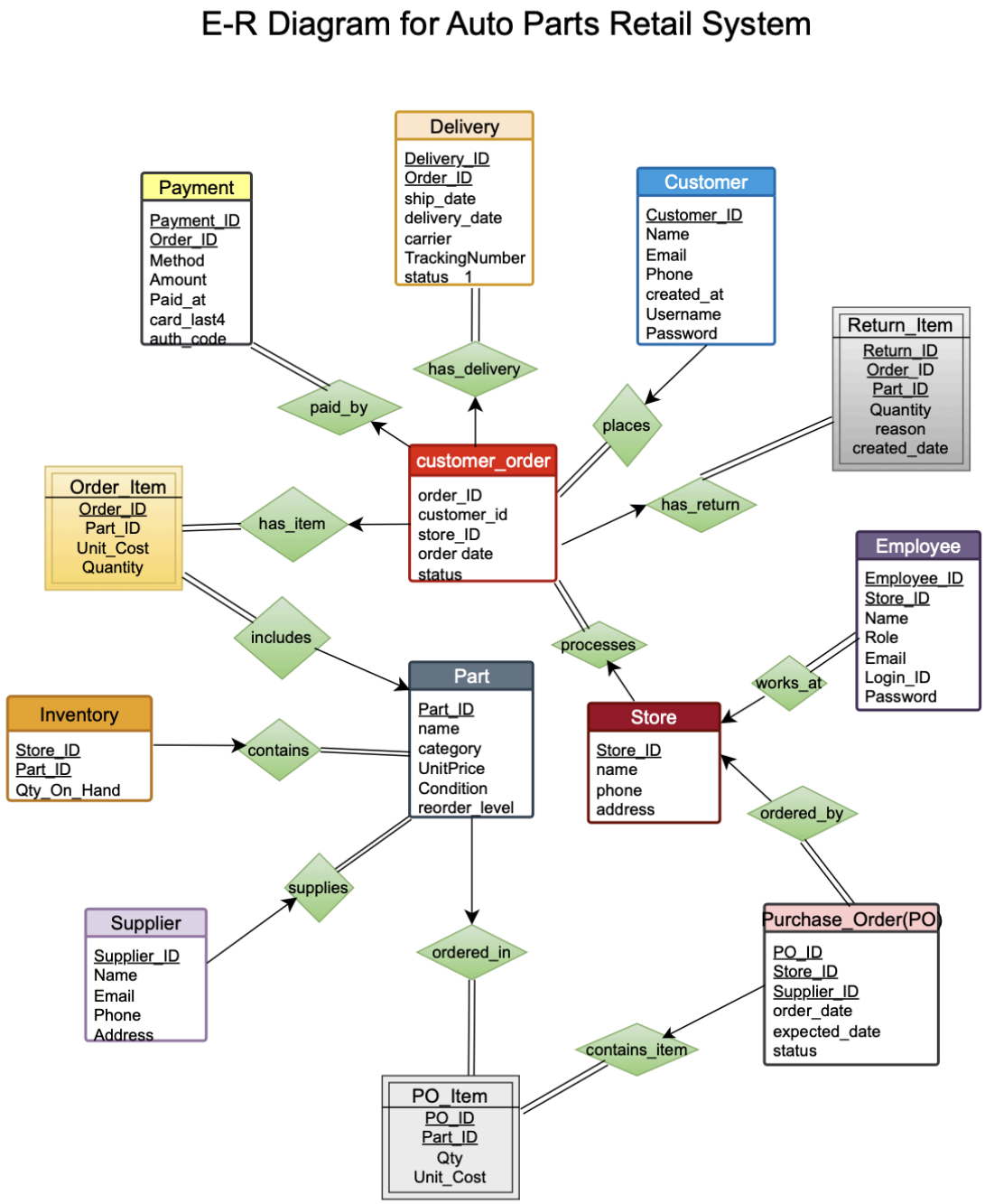


Figure1: E-R diagram

Relational Schema

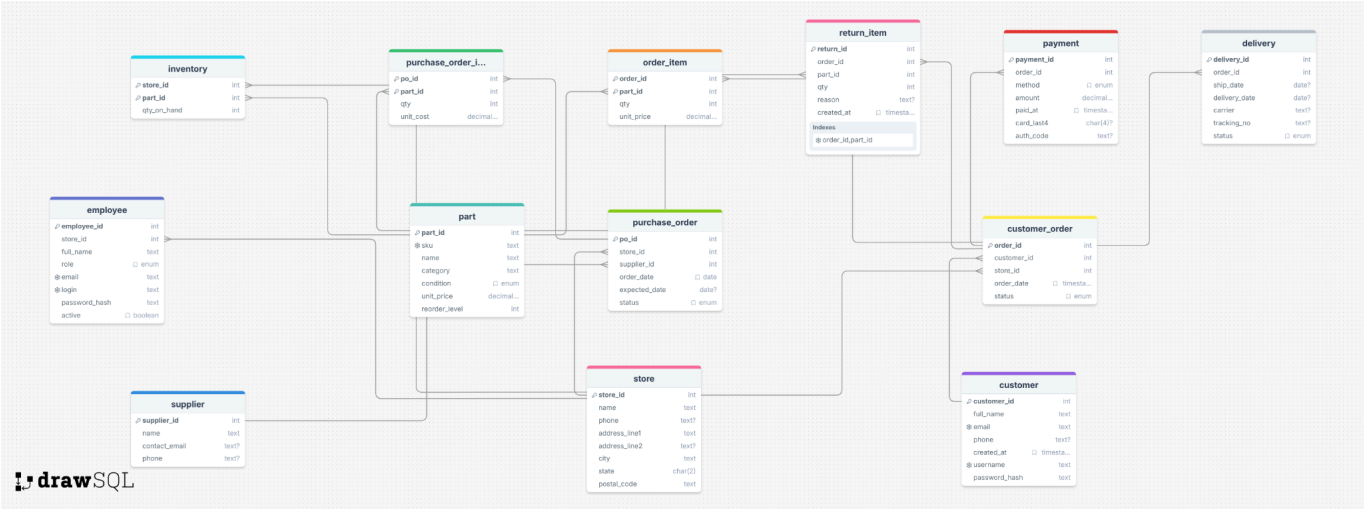
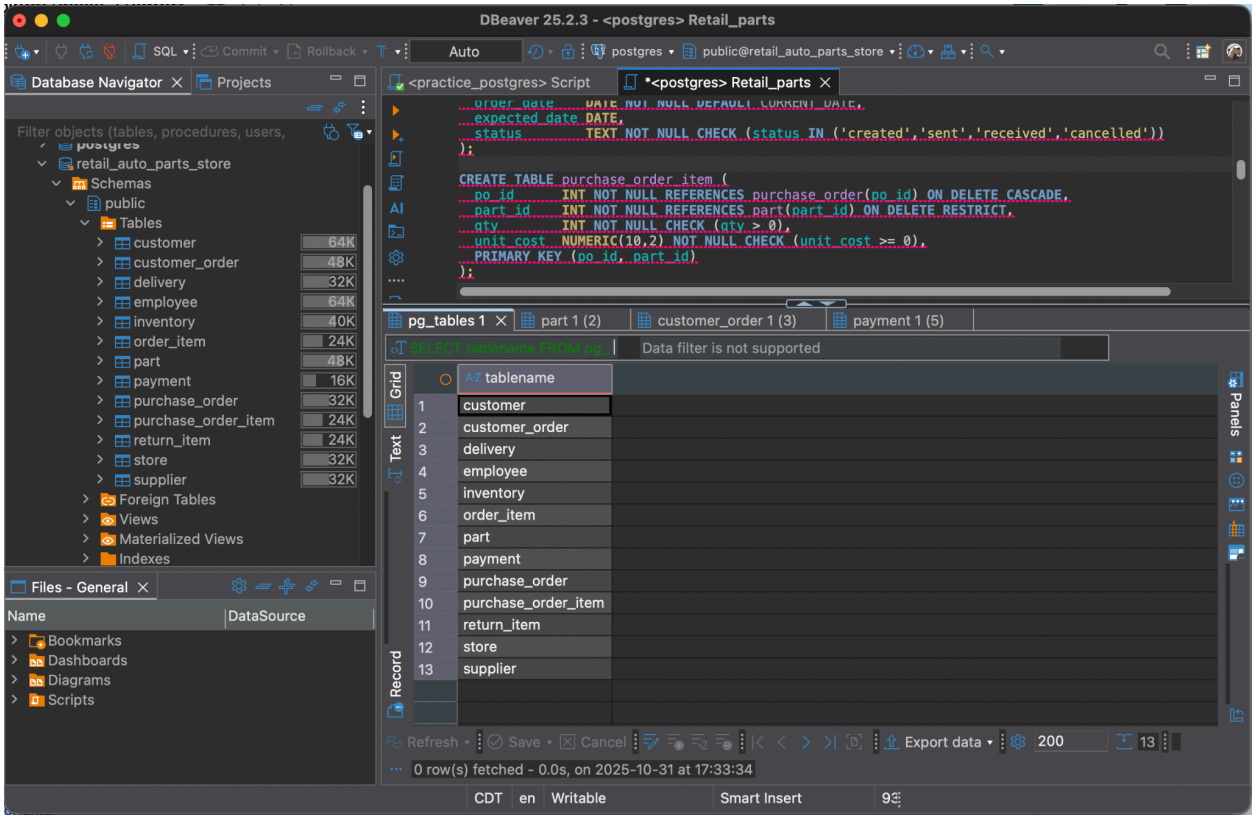


Figure 2: Relational Schema Diagram

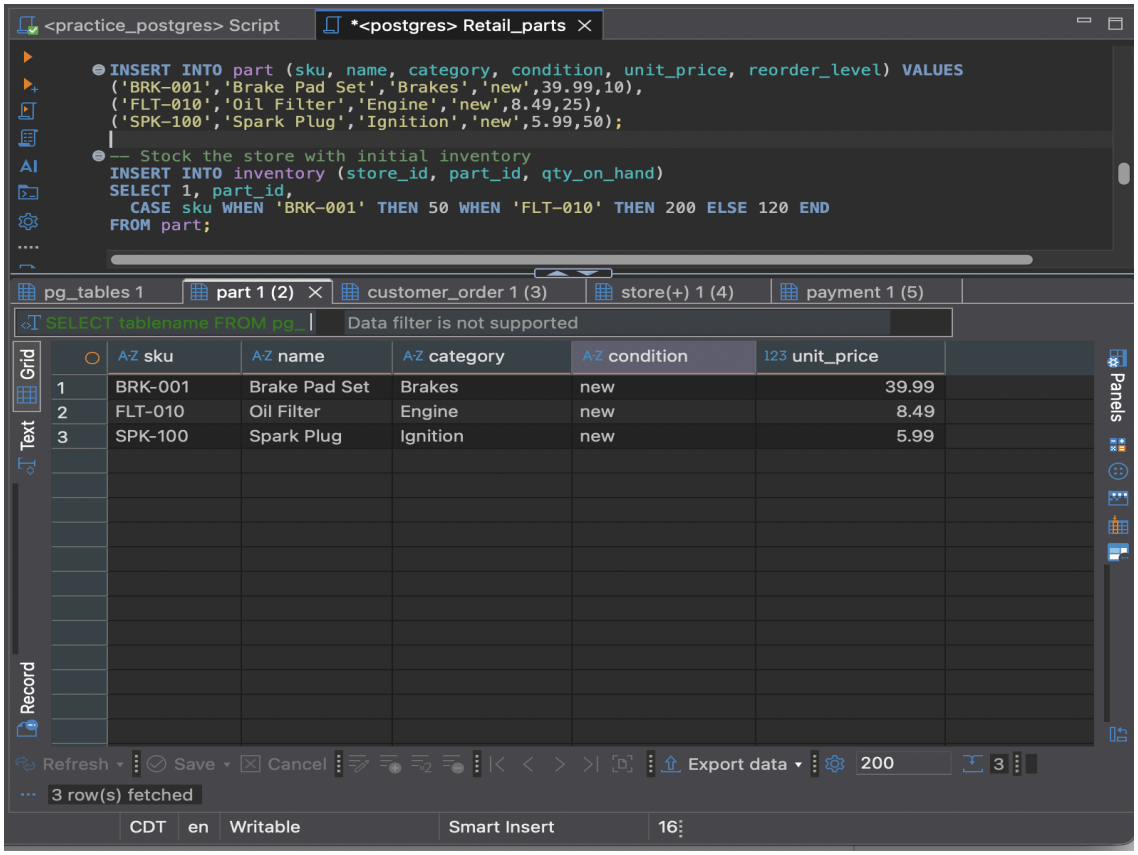
Sample Data and Output Results

CREATE TABLE



Parts table output

```
SELECT sku, name, category, condition, unit_price FROM part ORDER BY name;
```



Team Member Contribution

Quoc Nhan Tra

- Implemented tables in PostgreSQL
- Helped define entity, attributes, and datatypes
- Tested queries and provided samples with screenshots
- Aided in compiling Phase II report

Binod Chhantyal

- Drew ER diagram
- Designed relational schema
- Defined entity, attributes, and datatypes

Solomon Rosenfeld

- Validated normalization process
- aided in establishing referential integrity
- Compiled Phase II report

Conclusion

In phase II, we were able to successfully create an executable relational database schema. The database we created enforces strong integrity constraints, and supports major operations required by the Retail Auto Parts System. The ER and Schema diagrams illustrate a conceptual map for the logical model. Additionally, we were able to produce screenshots using sample data for execution to confirm the database is working within the scope of the designs and constraints. Lastly, improvements to planning the software architecture were made through identifying problems surfaced from practical testing and implementation.