LogiCore: A Relational Database for Optimizing U.S. Supply Chain and Logistics Operations

Henil Sureshbhai Diyora[1], Divyesh Sankhla[2], Sachi Ramaswamy[3]
Dept. of Computer Science and Engineering, University at Buffalo, NY, USA
UB ID: henilsur[1], divyeshk[2], sachiram[3]

***Abstract*—Modern supply chain operations suffer from significant inefficiencies due to data fragmentation across disparate systems. This project addresses the challenge by designing and implementing LogiCore, a centralized relational database for the U.S. logistics sector. The system integrates all key operational data including customers, products, inventory, purchase orders, and shipments into a single, cohesive schema. The primary objective is to create a robust data backbone that enables comprehensive analysis, real-time tracking, and data-driven decision making, thereby overcoming the limitations of traditional spreadsheet-based management. This paper details the problem statement, database design, and relational schema.**

***Index Terms*—Supply Chain Management, Logistics, Database Systems, PostgreSQL, Data Modeling, E/R Diagram, BCNF**

I. Introduction and Problem Statement

A. The Challenge of Data Fragmentation

Modern supply chain operations are characterized by immense complexity. The critical challenge lies in the **fragmentation of data**; information on customers, products, inventory, and shipments often exists in separate, unlinked systems. This leads to significant inefficiencies such as:

- **Poor Inventory Management:** Resulting in stockouts (lost sales) or overstocking (increased holding costs).
- **Inefficient Order Fulfillment:** Causing delays in processing and shipping orders.
- **Lack of Performance Insight:** Preventing data-driven evaluation of suppliers and shipping carriers.
- **Reactive Customer Service:** Hindering the ability to provide real-time, accurate updates.

This project aims to solve these issues by designing and implementing a centralized relational database that integrates all key aspects of the logistics lifecycle.

B. Project Objectives and Scope

The primary objective is to build a robust database that serves as a single source of truth. This system will be designed to answer complex questions critical for business intelligence, such as real-time inventory levels, top-selling products, and carrier performance metrics.

C. Justification for a Relational Database Approach

A relational database management system (RDBMS) like PostgreSQL is essential for this problem due to its strengths in:

- **Data Integrity:** Enforcing primary/foreign key constraints.
- **Scalability:** Handling millions of records efficiently.
- **Complex Queries:** Joining multiple tables to derive insight.
- **Concurrency:** Supporting multiuser operations safely.

D. Contribution to the Domain

This project's contribution is the creation of a blueprint for a unified logistics management system. By providing a centralized data backbone, this database empowers a business to move from a reactive to a proactive operational model, leading to cost reduction, improved customer satisfaction, and enhanced strategic decision-making.

II. Target Users and System Roles

The LogiCore database is designed as a central hub for various roles, from operational staff to strategic analysts.

A. User Profiles and Scenarios

1) The Logistics Analyst: A power user focused on business intelligence. *Scenario:* An analyst queries the `Shipments`, `ShippingCarriers`, and `Orders` tables using `GROUP BY` and `AVG(shipping_cost)` to rank carriers by cost-effectiveness and delivery speed.

2) The Warehouse Manager: An operational user responsible for inventory and fulfillment. *Scenario:* The manager monitors the `Inventory` table, checking `quantity_on_hand` against `reorder_level`. If stock is low, they initiate a new entry in the `PurchaseOrders` table.

3) Customer Service Representative (CSR): A front-line user who interacts directly with customers. *Scenario:* A customer inquires about their order. The CSR queries the `Customers` table, then joins `Shipments` and `ShipmentTrackingHistory` to provide up-to-the-minute package status.

B. System Administration

1) The Database Administrator (DBA): The technical guardian of the system, responsible for performance, integrity, and security. *Responsibilities:* The DBA manages roles, performs backups, and uses `EXPLAIN` to profile and optimize slow queries.

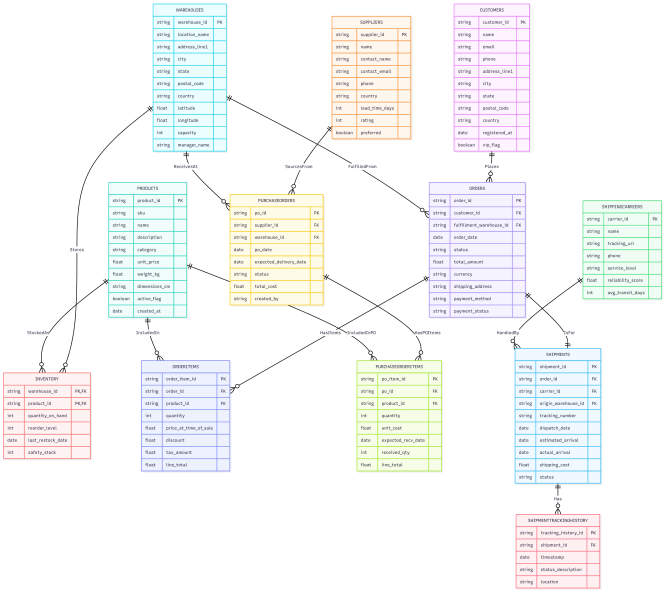

Fig. 1: Entity–relationship (E/R) diagram for the LogiCore database.

III. Database Schema Design (E/R Diagram)

The database schema was designed using an Entity–Relationship (E/R) model to visualize the entities, their attributes, and relationships. The complete E/R diagram for the LogiCore database is shown in Fig. 1.

The design includes 12 entities. Many-to-many (M:N) relationships are resolved using associative entities:

- The M:N relationship between `Orders` and `Products` is resolved by `OrderItems`.
- The M:N relationship between `Warehouses` and `Products` is resolved by `Inventory`.
- The M:N relationship between `PurchaseOrders` and `Products` is resolved by `PurchaseOrderItems`.

This design ensures all relationships are well-defined and the schema is normalized, preventing redundancy.

A. Cardinality and Participation Summary

Each relationship shown in Fig. 1 follows standard crow's-foot notation for cardinality. The relationships and their corresponding participation constraints are summarized below:

- **Customers–Orders:** One customer can place many orders (1:N). Participation is total on Orders.
- **Orders–OrderItems:** Each order contains one or more line items (1:N). Participation is total on OrderItems.
- **Warehouses–Products:** One warehouse stocks many products (1:N). Participation is total on Inventory.
- **Suppliers–PurchaseOrders:** One supplier can receive multiple purchase orders (1:N). Participation is total on PurchaseOrders.
- **PurchaseOrders–Products:** Each purchase order can include several products (1:N). Participation is total on PurchaseOrderItems.
- **Orders–Shipments:** Each order produces at most one shipment (1:1). Participation is partial on Shipments.
- **Shipments–ShippingCarriers:** Many shipments are handled by one carrier (N:1). Participation is total on Shipments.
- **Warehouses–Shipments:** A warehouse can originate many shipments (1:N). Participation is partial on Shipments.
- **Shipments–ShipmentTrackingHistory:** Each shipment can have multiple tracking events (1:N). Participation is total on TrackingHistory.

All entities in this schema are strong entities. Double-line participation (total) indicates that every dependent record must reference its parent, whereas single-line participation (partial) indicates an optional relationship.

B. E/R to Relational Mapping Summary

Each entity and relationship from the E/R model was systematically converted into a corresponding relation in the logical schema as follows:

- **Customers** → relation `Customers(customer_id, ...)`.
- **Products** → relation `Products(product_id, ...)`.
- **Suppliers** → relation `Suppliers(supplier_id, ...)`.
- **Warehouses** → relation `Warehouses(warehouse_id, ...)`.
- **ShippingCarriers** → relation `ShippingCarriers(carrier_id, ...)`.
- **Orders** → relation `Orders(order_id, customer_id, warehouse_id, ...)`.
- **Order–Product (M:N)** → relation `OrderItems(order_item_id, order_id, product_id, ...)`.
- **Warehouse–Product (M:N)** → relation `Inventory(warehouse_id, product_id, ...)`.
- **PurchaseOrders** → relation `PurchaseOrders(po_id, supplier_id, warehouse_id, ...)`.
- **PurchaseOrder–Product (M:N)** → relation `PurchaseOrderItems(po_item_id, po_id, product_id, ...)`.
- **Order–Shipment (1:1)** → relation `Shipments(shipment_id, order_id, carrier_id, warehouse_id, ...)`.
- **Shipment–TrackingHistory (1:N)** → relation `ShipmentTrackingHistory(tracking_history_id, shipment_id, ...)`.

This mapping ensures that all many-to-many relationships were properly decomposed into associative entities and that every relation has a clearly defined primary key and referential links.

IV. Relational Schema and BCNF Analysis

This section provides a detailed data dictionary for each relation and formal proof that each relation is in Boyce–Codd Normal Form (BCNF). A relation is in BCNF if, for every non-trivial functional dependency $X \to Y$, X is a superkey.

Our dataset was program-generated to ensure integrity and contains *10,000 records per table*, satisfying the graduate requirement (minimum 3,000). All 12 relations satisfy BCNF and are free from redundancy and anomalies.

A. Customers Table

- **Description:** Stores unique information for each customer.
- **Key Justification:** `customer_id` is a UUID, guaranteeing a unique primary key for every record.

TABLE I: Customers Table

Attribute	Data Type	Description / Constraints
customer_id	UUID	PK, NOT NULL. Unique identifier.
name	VARCHAR(255)	NOT NULL. Customer's full name.
email	VARCHAR(255)	UNIQUE, NOT NULL. Customer's email.
phone	VARCHAR(50)	NULLable. Phone number.
address_line1	VARCHAR(255)	NULLable. Street address.
city	VARCHAR(100)	NULLable. City.
state	VARCHAR(50)	NULLable. State (US Abbr.).
postal_code	VARCHAR(20)	NULLable. Postal code.
country	VARCHAR(50)	NULLable. Country (default 'USA').
registered_at	TIMESTAMPTZ	NULLable. Date of registration.
vip_flag	BOOLEAN	NULLable. Flag for VIP status.

- **Foreign Key Policies:** This table has no foreign keys.
- **Functional Dependencies (FDs):**
  1) {customer_id} → {name, email, phone, ...}
  2) {email} → {customer_id, name, phone, ...}
- **BCNF Analysis:** The determinants are {customer_id} (Primary Key) and {email} (Candidate Key). Since both are superkeys, the **Customers** relation is in BCNF.

B. Products Table

- **Description:** Stores all product catalog information.
- **Key Justification:** `product_id` (UUID) is the primary key. `sku` is also a unique identifier.

TABLE II: Products Table

Attribute	Data Type	Description / Constraints
product_id	UUID	PK, NOT NULL. Unique identifier.
sku	VARCHAR(100)	UNIQUE, NOT NULL. Stock Keeping Unit.
name	VARCHAR(255)	NOT NULL. Product name.
description	TEXT	NULLable. Product description.
category	VARCHAR(100)	NULLable. Product category.
unit_price	NUMERIC(10,2)	NOT NULL. Price per unit.
weight_kg	NUMERIC(10,3)	NULLable. Product weight.
dimensions_cm	VARCHAR(100)	NULLable. Product dimensions.
active_flag	BOOLEAN	NULLable. If product is active.
created_at	TIMESTAMPTZ	NULLable. Date product was added.

- **Foreign Key Policies:** This table has no foreign keys.
- **Functional Dependencies (FDs):**
  1) {product_id} → {sku, name, description, ...}
  2) {sku} → {product_id, name, description, ...}
- **BCNF Analysis:** The determinants are {product_id} (PK) and {sku} (Candidate Key). Both are superkeys. Thus, the **Products** relation is in BCNF.

C. Suppliers Table

- **Description:** Stores information about product suppliers.
- **Key Justification:** `supplier_id` (UUID) is the primary key.

TABLE III: Suppliers Table

Attribute	Data Type	Description / Constraints
supplier_id	UUID	PK, NOT NULL. Unique identifier.
name	VARCHAR(255)	NOT NULL. Supplier company name.
contact_name	VARCHAR(255)	NULLable. Contact person's name.
contact_email	VARCHAR(255)	NULLable. Contact email.
phone	VARCHAR(50)	NULLable. Phone number.
country	VARCHAR(100)	NULLable. Country (default 'USA').
lead_time_days	INTEGER	NULLable. Avg. lead time.
rating	NUMERIC(3,2)	NULLable. Supplier rating.
preferred	BOOLEAN	NULLable. Preferred supplier flag.

- **Foreign Key Policies:** This table has no foreign keys.
- **FDs:** {supplier_id} → {name, contact_name, ...}
- **BCNF Analysis:** The only determinant is {supplier_id}, which is the primary key and thus a superkey. The **Suppliers** relation is in BCNF.

D. Warehouses Table

- **Description:** Stores details of physical warehouse locations.
- **Key Justification:** `warehouse_id` (UUID) is the primary key.

TABLE IV: Warehouses Table

Attribute	Data Type	Description / Constraints
warehouse_id	UUID	PK, NOT NULL. Unique identifier.
location_name	VARCHAR(255)	NOT NULL. e.g., "Chicago Facility".
address_line1	VARCHAR(255)	NULLable. Street address.
city	VARCHAR(100)	NULLable. City.
state	VARCHAR(100)	NULLable. State.
postal_code	VARCHAR(20)	NULLable. Postal code.
country	VARCHAR(100)	NULLable. Country.
latitude	NUMERIC(9,6)	NULLable. GPS coordinate.
longitude	NUMERIC(9,6)	NULLable. GPS coordinate.
capacity	INTEGER	NULLable. Storage capacity.
manager_name	VARCHAR(255)	NULLable. Manager's name.

- **Foreign Key Policies:** This table has no foreign keys.
- **FDs:** {warehouse_id} → {location_name, city, ...}
- **BCNF Analysis:** The only determinant is the primary key {warehouse_id}. Thus, the **Warehouses** relation is in BCNF.

E. ShippingCarriers Table

- **Description:** Stores information on shipping partners.
- **Key Justification:** carrier_id (UUID) is the primary key.

TABLE V: Shipping Carriers Table

Attribute	Data Type	Description / Constraints
carrier_id	UUID	PK, NOT NULL. Unique identifier.
name	VARCHAR(255)	NOT NULL. Carrier name (e.g., "FedEx").
tracking_url	VARCHAR(512)	NULLable. URL prefix for tracking.
phone	VARCHAR(50)	NULLable. Contact phone.
service_level	VARCHAR(50)	NULLable. (e.g., "Standard", "Express").
reliability_score	NUMERIC(4,3)	NULLable. A performance metric.
avg_transit_days	INTEGER	NULLable. Avg. delivery time.

- **Foreign Key Policies:** This table has no foreign keys.
- **FDs:** {carrier_id} → {name, tracking_url, ...}
- **BCNF Analysis:** The only determinant is the primary key {carrier_id}. Thus, the **ShippingCarriers** relation is in BCNF.

F. Orders Table

- **Description:** Stores the header information for each customer order.
- **Key Justification:** order_id (UUID) is the primary key.

TABLE VI: Orders Table

Attribute	Data Type	Description / Constraints
order_id	UUID	PK, NOT NULL. Unique identifier.
customer_id	UUID	FK, NOT NULL. Refs Customers.
order_date	TIMESTAMPTZ	NULLable. Time order was placed.
status	VARCHAR(50)	NULLable. (e.g., "Placed", "Shipped").
total_amount	NUMERIC(12,2)	NULLable. Total cost of order.
currency	VARCHAR(10)	NULLable. (e.g., "USD").
shipping_address	TEXT	NULLable. Full shipping address.
payment_method	VARCHAR(50)	NULLable. (e.g., "Credit Card").
payment_status	VARCHAR(50)	NULLable. (e.g., "Paid", "Pending").
fulfillment_warehouse_id	UUID	FK, NULLable. Refs Warehouses.

- **Foreign Key Policies:**
  - customer_id references Customers. We use ON DELETE NO ACTION (default) to prevent deleting a customer who has existing orders, ensuring historical integrity.
  - fulfillment_warehouse_id references Warehouses. ON DELETE NO ACTION prevents deleting a warehouse if it is tied to orders.
- **FDs:** {order_id} → {customer_id, order_date, ...}
- **BCNF Analysis:** The only determinant is the primary key {order_id}. Thus, the **Orders** relation is in BCNF.

G. Inventory Table (Associative)

- **Description:** Associative entity linking Warehouses and Products to track stock levels.
- **Key Justification:** A composite primary key {warehouse_id, product_id} ensures only one record for each unique product in each unique warehouse.

TABLE VII: Inventory Table

Attribute	Data Type	Description / Constraints
warehouse_id	UUID	PK, FK, NOT NULL. Refs Warehouses.
product_id	UUID	PK, FK, NOT NULL. Refs Products.
quantity_on_hand	INTEGER	NOT NULL. Stock on hand.
reorder_level	INTEGER	NULLable. Threshold to reorder.
last_restock_date	TIMESTAMPTZ	NULLable. Date of last restock.
safety_stock	INTEGER	NULLable. Buffer stock level.

- **Foreign Key Policies:**
  - Both warehouse_id and product_id use ON DELETE NO ACTION. We would not want to delete a warehouse or product if an inventory record still exists.
- **FDs:** {warehouse_id, product_id} → {quantity_on_hand, ...}
- **BCNF Analysis:** The only determinant is the composite key {warehouse_id, product_id}, which is a superkey. Thus, the **Inventory** relation is in BCNF.

H. PurchaseOrders Table

- **Description:** Stores header information for purchase orders sent to suppliers.
- **Key Justification:** po_id (UUID) is the primary key.
- **Foreign Key Policies:**
  - Both supplier_id and warehouse_id use ON DELETE NO ACTION to maintain historical records of purchase orders, even if a supplier or warehouse is later removed from active status.
- **FDs:** {po_id} → {supplier_id, warehouse_id, ...}
- **BCNF Analysis:** The only determinant is the primary key {po_id}. Thus, the **PurchaseOrders** relation is in BCNF.

TABLE VIII: Purchase Orders Table

Attribute	Data Type	Description / Constraints
po_id	UUID	PK, NOT NULL. Unique identifier.
supplier_id	UUID	FK, NOT NULL. Refs Suppliers.
warehouse_id	UUID	FK, NOT NULL. Refs Warehouses.
po_date	TIMESTAMPTZ	NULLable. Date PO was created.
expected_delivery_date	DATE	NULLable. When delivery is expected.
status	VARCHAR(50)	NULLable. (e.g., "Ordered", "Received").
total_cost	NUMERIC(12,2)	NULLable. Total cost of PO.
created_by	VARCHAR(255)	NULLable. Employee who created PO.

I. OrderItems Table (Associative)

- **Description:** Associative entity linking Orders and Products. Stores the line items for each customer order.
- **Key Justification:** order_item_id (UUID) is a surrogate primary key.

TABLE IX: Order Items Table

Attribute	Data Type	Description / Constraints
order_item_id	UUID	PK, NOT NULL. Unique identifier.
order_id	UUID	FK, NOT NULL. Refs Orders.
product_id	UUID	FK, NOT NULL. Refs Products.
quantity	INTEGER	NOT NULL. Quantity ordered.
price_at_time_of_sale	NUMERIC(10,2)	NOT NULL. Price when sold.
discount	NUMERIC(4,2)	NULLable. Discount applied.
tax_amount	NUMERIC(10,2)	NULLable. Tax for this item.
line_total	NUMERIC(12,2)	NULLable. (Quantity * Price) - Discount.

- **Foreign Key Policies:**
  - order_id references Orders. We use ON DELETE CASCADE, so if an order is deleted, all its associated line items are also deleted.
  - product_id references Products. ON DELETE NO ACTION is safer to prevent deleting a product that exists in past orders.
- **FDs:** {order_item_id} → {order_id, product_id, ...}
- **BCNF Analysis:** The only determinant is the primary key {order_item_id}. Thus, the **OrderItems** relation is in BCNF.

J. PurchaseOrderItems Table (Associative)

- **Description:** Associative entity linking PurchaseOrders and Products. Stores line items for each PO.
- **Key Justification:** po_item_id (UUID) is a surrogate primary key.

TABLE X: Purchase Order Items Table

Attribute	Data Type	Description / Constraints
po_item_id	UUID	PK, NOT NULL. Unique identifier.
po_id	UUID	FK, NOT NULL. Refs PurchaseOrders.
product_id	UUID	FK, NOT NULL. Refs Products.
quantity	INTEGER	NOT NULL. Quantity ordered.
unit_cost	NUMERIC(10,2)	NULLable. Cost per unit.
expected_recv_date	DATE	NULLable. Expected arrival.
received_qty	INTEGER	NULLable. Quantity actually received.
line_total	NUMERIC(12,2)	NULLable. (Quantity * UnitCost).

- **Foreign Key Policies:**
  - po_id references PurchaseOrders. We use ON DELETE CASCADE, as line items are useless without their parent PO.
  - product_id references Products. ON DELETE NO ACTION protects historical data.
- **FDs:** {po_item_id} → {po_id, product_id, ...}
- **BCNF Analysis:** The only determinant is the primary key {po_item_id}. Thus, the **PurchaseOrderItems** relation is in BCNF.

K. Shipments Table

- **Description:** Stores information about the physical shipment of a customer's order.
- **Key Justification:** shipment_id (UUID) is the primary key. tracking_number is also a unique key.

TABLE XI: Shipments Table

Attribute	Data Type	Description / Constraints
shipment_id	UUID	PK, NOT NULL. Unique identifier.
order_id	UUID	FK, NOT NULL. Refs Orders.
carrier_id	UUID	FK, NOT NULL. Refs ShippingCarriers.
tracking_number	VARCHAR(255)	UNIQUE, NULLable. Tracking code.
dispatch_date	TIMESTAMPTZ	NULLable. Time shipped.
estimated_arrival	DATE	NULLable. ETA.
actual_arrival	TIMESTAMPTZ	NULLable. Time delivered.
shipping_cost	NUMERIC(10,2)	NULLable. Cost of shipment.
origin_warehouse_id	UUID	FK, NULLable. Refs Warehouses.
status	VARCHAR(50)	NULLable. (e.g., "In Transit").

- **Foreign Key Policies:**
  - All FKs (order_id, carrier_id, origin_warehouse_id) use ON DELETE NO ACTION to preserve historical shipment records, even if the referenced order or carrier is removed.

- **Functional Dependencies (FDs):**
  1) {shipment_id} → {order_id, carrier_id, ...}
  2) {tracking_number} → {shipment_id, order_id, ...}
- **BCNF Analysis:** The determinants are {shipment_id} (PK) and {tracking_number} (Candidate Key). Both are superkeys. Thus, the **Shipments** relation is in BCNF.

L. ShipmentTrackingHistory Table

- **Description:** Stores the time-series event data for each shipment (e.g., "In Transit," "Out for Delivery").
- **Key Justification:** tracking_history_id (UUID) is the primary key.

TABLE XII: Shipment Tracking History Table

Attribute	Data Type	Description / Constraints
tracking_history_id	UUID	PK, NOT NULL. Unique identifier.
shipment_id	UUID	FK, NOT NULL. Refs Shipments.
timestamp	TIMESTAMPTZ	NULLable. Time of the event.
status_description	VARCHAR(255)	NULLable. Description of the event.
location	VARCHAR(255)	NULLable. Location of the event.

- **Foreign Key Policies:**
  - shipment_id references Shipments. This uses ON DELETE CASCADE. If a shipment record is deleted, its tracking history is no longer relevant and should be deleted automatically.
- **FDs:** {tracking_history_id} → {shipment_id, timestamp, ...}
- **BCNF Analysis:** The only determinant is the primary key {tracking_history_id}. Thus, the **ShipmentTrackingHistory** relation is in BCNF.

M. Example: Attribute Closure and Key Validation

To demonstrate BCNF verification, consider the Shipments relation.

Given functional dependencies:

- (1) {shipment_id} → {order_id, carrier_id, tracking_number, status, shipping_cost}
- (2) {tracking_number} → {shipment_id, order_id, carrier_id, status, shipping_cost}

Closure computation:

$(\texttt{shipment\_id})^+ = \{\texttt{shipment\_id, order\_id, carrier\_id, tracking\_number, status, shipping\_cost}\}$

Because this closure includes every attribute of the relation, shipment_id is a superkey. Similarly, tracking_number determines all attributes, proving it is a candidate key. Hence, all determinants in this relation are superkeys, confirming that Shipments satisfies BCNF.

N. Lossless Join and Dependency Preservation

Every decomposition in the LogiCore schema satisfies the *lossless-join property* because the intersecting attribute in each decomposition is a primary key or candidate key of one of the relations. This ensures that when decomposed relations are joined back together, no spurious tuples are produced and no information is lost.

Furthermore, all decompositions are *dependency preserving*. Each functional dependency can be enforced within at least one relation without the need to recombine relations through joins. This guarantees both correctness and efficiency during data updates and constraint enforcement.

O. Normalization Conclusion

All 12 relations in the LogiCore schema were formally examined. Every non-trivial functional dependency has a determinant that is a superkey. Hence, the entire database schema is in *Boyce–Codd Normal Form (BCNF)*, and no further decomposition is required.

V. Conclusion

This paper presented *LogiCore*, a relational database system engineered for modern U.S. logistics operations. By integrating disparate datasets—customers, products, warehouses, and carriers—into a unified model, the system provides a foundation for real-time visibility, analytical reporting, and operational efficiency. The design, validated through BCNF normalization, ensures scalability, consistency, and strong referential integrity for future analytics and decision support systems.

Acknowledgment

The authors would like to thank the *Department of Computer Science and Engineering*, University at Buffalo, for the guidance and resources provided during this project.

References

[1] Faker Documentation, "Python Faker Library," *Read the Docs*, 2025. [Online]. Available: https://faker.readthedocs.io/en/master/. [Accessed: Oct. 20, 2025].

[2] MermaidChart, "Mermaid Chart Diagramming Tool," *MermaidChart.com*, 2025. [Online]. Available: https://www.mermaidchart.com/app/dashboard. [Accessed: Oct. 20, 2025].