E-Commerce (Target 2.0) Management System

Project Team3:

Yang Zheng 002771462

Mengyun Xie 002754435

Chenni Xu 002640478

Overview

The Target 2.0 E-Commerce Management System will redefine the online shopping experience through the seamless management of Customers, orders, products, Coupons, and logistics. The system ensures seamless operations with much precision, security, and in real time for both customers and administrators. It creates trust among customers, makes life easier, provides scalability, and thereby enables business growth, innovation, and great digital retail experiences.

Business Problems and Objectives

- Customer Management
  - Maintain (enter, update, and delete) Customer profiles, including email addresses, phone numbers, and password-protected accounts.
  - Manage Customer addresses, supporting multiple addresses with default options for ease of checkout.
  - Track Customer payment methods, including details like payment type, provider, account number, and expiration dates.
- Product Management
  - Maintain data on products, including descriptions, categories, pricing, images, and SKU details.
  - Track product variations, including size, color, and other configurable options.
  - Manage product categories and subcategories for better catalog organization.
  - Monitor stock availability to prevent overselling.
- Shopping Cart and Order Management
  - Enable the creation and management of shopping carts for Customers, tracking selected products and their quantities.
  - Record and manage orders, including payment methods, shipping methods, addresses, and order totals.
  - Track the status of orders throughout their lifecycle (e.g., pending, shipped, delivered, returned).

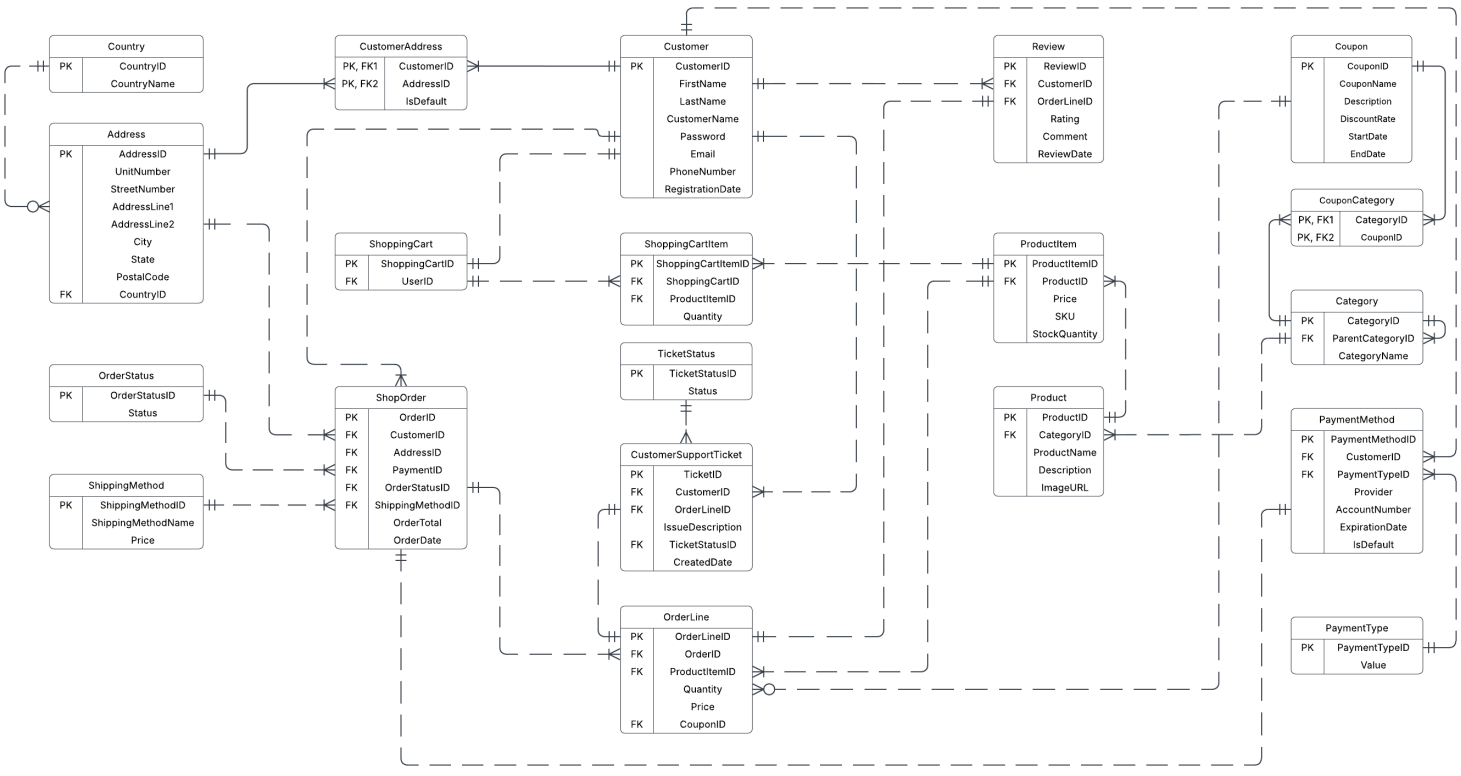
- Order Line Management
  - Track details of each item in an order, including quantities and item-specific pricing.
- Coupons
  - Maintain data on Coupons, including descriptions, discount rates, and validity periods.
  - Associate Coupons with specific product categories or items for targeted marketing.
- Review and Feedback Management
  - Enable Customers to provide reviews and ratings for ordered products, along with optional comments for feedback.
- Customer support:
  - Implement a ticketing system to enable customers to raise queries, report issues, and request assistance.
  - Provide automated updates on ticket status (e.g., open, in progress, resolved).
- Data Security and Integrity
  - Ensure secure handling of Customer data, including passwords and payment information.
  - Maintain historical records for auditing purposes while providing mechanisms for updates and deletions when needed.

Business Rules

A Customer can have multiple addresses.
A Customer can have multiple payment methods.
A Customer can have a shopping cart.
A Customer can submit multiple reviews.
A Customer can create multiple tickets.
A Customer can have multiple shop orders.
An address can be used for multiple orders.
A country can have multiple addresses.
An order can belong to one payment method.
An order contains multiple order lines.
An order has an order status.
An order status has many corresponds orders.
A Payment type can belong to multiple payment methods.
A product belongs to a category.
A product can have many product items.

A category contains multiple products.
Categories can have subcategories.
A category can have multiple coupons.
A product item belongs to multiple order lines.
A product item can link to many shopping carts.
A shopping cart can contain multiple items.
A Coupon can be used for multiple order line.
A product item can be multiple shopping cart items.
A review belongs to an order line.
A ticket status can belong to many tickets.
A Shipping Method can be used for multiple orders.

E-R Diagram



Note: If the embedded E-R diagram appears too small to read, please refer to 03.Final_ERD_Team3.png. located in the same package.

Entities & Relationships & Design Decision

1. Customer

- **Attributes:**
  - CustomerID (Primary Key)
  - FirstName
  - LastName
  - CustomerName
  - Email
  - PhoneNumber
  - Password (hashed)
  - RegistrationDate
- **Relationships:**
  - One-to-Many with Customer Address (Each Customer can have multiple addresses)
  - One-to-Many with Payment Method (A Customer can have multiple payment methods)
  - One-to-One with Shopping Cart (Each Customer can have a shopping cart)
  - One-to-Many with Review (Each Customer can submit multiple reviews)
  - One-to-Many with Customer Support Ticket (Each Customer can create multiple tickets)
- **Why Included:** Central to the system, Customers interact with the platform for shopping, managing orders, making payments, and contacting support. Essential for personalized experiences like account management and activity tracking.

2. CustomerAddress

- **Attributes:**
  - AddressID (Primary Key, Foreign Key)
  - CustomerID (Primary Key, Foreign Key)
  - IsDefault (Boolean)
- **Relationships:**
  - Many-to-One with Customer (Each Customer can have multiple Customer addresses)
  - Many-to-One with Address (Each address can have multiple Customer addresses)
- **Why Included:** Allows Customers to save multiple addresses (e.g., home, office) for convenience. Differentiates between Customers and shared addresses, ensuring reusability.

3. Address

- **Attributes:**
  - AddressID (Primary Key)
  - UnitNumber
  - StreetNumber
  - AddressLine1
  - AddressLine2
  - City
  - State
  - ZipCode
  - CountryID (Foreign Key)
- **Relationships:**
  - One-to-Many with Order (An address can be used for multiple orders)
  - Many-to-One with Country (Each country can have multiple addresses)
  - One-to-Many with Customer Address (Each address can have multiple Customer addresses)
- **Why Included:** Facilitates address reuse across multiple Customers, orders, or scenarios. Stores detailed information about location and is linked to countries for accuracy.

4. Country

- **Attributes:**
  - CountryID (Primary Key)
  - CountryName
- **Relationships:**
  - One-to-Many with address (Each country can have multiple addresses)
- **Why Included:** Stores country details to support international transactions and address normalization.

5. PaymentMethod

- **Attributes:**
  - PaymentMethodID (Primary Key)
  - CustomerID (Foreign Key)
  - PaymentType (Foreign Key, e.g., Credit Card, PayPal)
  - Provider (e.g., Visa, MasterCard)
  - AccountNumber (Masked)

- ExpirationDate
- IsDefault (Boolean)

- **Relationships:**
  - Many-to-One with Customer (A Customer can have multiple payment methods)
  - Many-to-One with Payment Type (A Payment type can belong to multiple Customer payment methods)
  - One-to-One with Order (An order can belong to one payment method)

- **Why Included:** Enables Customers to securely save multiple payment methods (credit cards, PayPal, etc.) for seamless checkout experiences.

6. PaymentType

- **Attributes:**
  - PaymentTypeID (Primary Key)
  - Value

- **Relationships:**
  - One-to-Many with Payment Method (A Payment type can belong to multiple payment methods)

- **Why Included:** Standardizes payment method categories (e.g., credit card, PayPal) for better organization and compatibility.

7. Product

- **Attributes:**
  - ProductID (Primary Key)
  - ProductName
  - Description
  - CategoryID (Foreign Key)
  - ImageURL

- **Relationships:**
  - Many-to-One with Category (Each product belongs to a category)
  - One-to-Many with Product Item (A product can have many product items)

- **Why Included:** Represents the core items being sold on the platform. Helps store basic details like name, description, and associated category for easier browsing.

8. Category

- **Attributes:**
  - CategoryID (Primary Key)
  - CategoryName

- ParentCategoryID (Foreign Key, Nullable)

- **Relationships:**
  - One-to-Many with Product (Each category contains multiple products)
  - Self-referential (Categories can have subcategories)
  - One-to-Many with Coupon category (Each category can have multiple Coupon category)

- **Why Included:** Groups products for easier navigation and search. Supports hierarchical categorization for subcategories (e.g., Electronics > Laptops).

9. ProductItem

- **Attributes:**
  - ProductItemID (Primary Key)
  - ProductID (Foreign Key)
  - Price
  - SKU
  - StockQuantity

- **Relationships:**
  - One-to-Many with order line (Each product item belongs to multiple order lines)
  - One-to-Many with shopping cart item (Each product item can linked to many shopping carts)

- **Why Included:** Tracks individual items, including their price, SKU, and stock, allowing for detailed inventory management.

10. CouponCategory

- **Attributes:**
  - CategoryID (Primary Key, Foreign Key)
  - CouponID (Primary Key, Foreign Key)

- **Relationships:**
  - Many-to-One with Category (Each category can have multiple Coupon categories)
  - Many-to-One with Coupon (Each Coupon can have multiple Coupon categories)

- **Why Included:** Associates Coupons with specific product categories, ensuring discounts are applied appropriately during sales events.

11. Coupon

- **Attributes:**
  - CouponID (Primary Key)
  - CouponName

- CouponDescription
- DiscountRate
- StartDate
- EndDate

- **Relationships:**
  - One-to-Many with Coupon Category (Each Coupon can have multiple Coupon categories)
  - One-to-Many with OrderLine (Each Coupon can be used for multiple OrderLine)
- **Why Included:** Represents sales campaigns or discounts applied to products or categories, improving marketing flexibility.

12. ShoppingCart

- **Attributes:**
  - CartID (Primary Key)
  - CustomerID (Foreign Key)
- **Relationships:**
  - One-to-One with Customer (Each Customer has one shopping cart)
  - One-to-Many with Shopping Cart Item (A cart can contain multiple items)
- **Why Included:** Essential for enabling Customers to temporarily save items before purchasing, simplifying the shopping process.

13. ShoppingCartItem

- **Attributes:**
  - ShoppingCartItemID (Primary Key)
  - ShoppingCartID (Foreign Key)
  - ProductID (Foreign Key)
  - Quantity
- **Relationships:**
  - Many-to-One with Shopping Cart (A shopping cart contains multiple items)
  - Many-to-One with Product Item (A product item can be multiple shopping cart item)
- **Why Included:** Tracks individual items in a Customer's shopping cart, including quantity and linked product details.

14. ShopOrder

- **Attributes:**
  - OrderID (Primary Key)

- CustomerID (Foreign Key)
- AddressID (Foreign Key)
- PaymentMethodID (Foreign Key)
- OrderTotal
- OrderStatus (Foreign Key, Pending, Shipped, Delivered, Returned)
- OrderDate

- **Relationships:**
  - Many-to-One with Address (Each address can have multiple shop orders)
  - Many-to-One with Customer (Each Customer can have multiple shop orders)
  - One-to-One with Payment Method (An order can only have one Customer payment method)
  - One-to-Many with Order Line (An order contains multiple order lines)
  - Many-to-One with Order Status (Each order have an order status)
- **Why Included:** Captures detailed order information, including payment, Customer, address, and status, for transaction tracking and fulfillment.

15. OrderLine

- **Attributes:**
  - OrderLineID (Primary Key)
  - OrderID (Foreign Key)
  - ProductItemID (Foreign Key)
  - CouponID (Foreign Key)
  - Quantity
  - Price (at the time of order)
- **Relationships:**
  - Many-to-One with Shop Order (An order contains multiple order lines)
  - Many-to-One with Product Item(Each product item corresponds to many order lines)
  - One-to-One with Review (An order line can have one review)
  - One-to-One with Customer Support Ticket (If applicable, an order line only corresponds to one review)
  - One-to-One with Coupon (If applicable, an order line only corresponds to one Coupon)
- **Why Included:** Tracks individual items in an order, including their quantity and price at purchase time. Critical for accurate order summaries.

16. OrderStatus

- **Attributes:**
  - OrderStatusID (Primary Key)
  - Status
- **Relationships:**
  - One-to-Many with Order (Each order status have many corresponds orders)
- **Why Included:** Represents the current state of an order (e.g., Pending, Shipped), improving tracking for both Customers and administrators.

17. Review

- **Attributes:**
  - ReviewID (Primary Key)
  - CustomerID (Foreign Key)
  - OrderLineID (Foreign Key)
  - Rating (1-5)
  - Comment
  - ReviewDate
- **Relationships:**
  - Many-to-One with Customer (Multiple reviews are written by a Customer)
  - One-to-One with Order line (Each review belongs to an Order line)
- **Why Included:** Allows Customers to provide feedback on purchased items, improving product quality and Customer engagement.

18. CustomerSupportTicket

- **Attributes:**
  - TicketID (Primary Key)
  - CustomerID (Foreign Key)
  - OrderLineID (Foreign Key, Nullable)
  - IssueDescription
  - TicketStatusID (Foreign Key, Open, In Progress, Resolved)
  - CreatedDate
- **Relationships:**
  - Many-to-One with Customer (Each ticket is created by a Customer)
  - One-to-One with Order line (If applicable, a ticket can be assigned to an Order line)

- **Why Included:** Tracks Customer-reported issues with orders or other platform experiences, enabling structured customer support.

19. TicketStatus:

- **Attributes:**
  - TicketStatusID (Primary Key)
  - Status
- **Relationships:**
  - One-to-Many with Ticket Status (Each ticket status can belong to many Customer Support Ticket)
- **Why Included:** Provides clear visibility into the status of support tickets (e.g., Open, Resolved), improving communication and resolution workflows.

20. ShippingMethod

- **Attributes:**
  - ShippingMethodID (Primary Key)
  - ShippingMethodName
  - Price
- **Relationships:**
  - One-to-Many with address (Each ShippingMethod can be used for multiple Order)
- **Why Included:** Stores ShippingMethod details to support various shipping methods.

Database Implementation

1. DDL.sql – Schema Definition

Purpose: Defines the complete database schema for the Target 2.0 E-Commerce Management System.

Implementation Details:

- Creates a database named E_COMMERCE.
- Defines 20 tables according to the E-R diagram.
- Uses appropriate SQL data types and integrity constraints.
- Implements **primary and foreign key constraints** to ensure referential integrity.
- Includes **simple and composite primary keys** (e.g., in the CustomerAddress table).
- Supports **hierarchical categorization** via self-referencing relationships in the Category table.

2. ComputedColumn.sql – Computed Columns

Purpose: Implements functions that provide calculated values based on existing data to simplify querying and maintain consistency.

Implemented Functions:

- GetCustomerFullName(CustomerID) – Returns a customer's full name.
- GetCustomerFormattedAddress(CustomerID) – Formats the customer's default address.
- GetTotalStock(ProductID) – Calculates total stock quantity for a product.
- GetStockStatus(ProductID) – Returns stock status like "In Stock", "Low Stock", or "Out of Stock".
- GetTop5BestSellers() – Returns a table of top 5 best-selling products.
- GetOrderGrandTotal(OrderID) – Computes grand total for an order with coupon discounts.
- GetUnsolvedTickets() – Returns a table of unresolved customer support tickets.

3. ColumnDataEncryption.sql – Sensitive Data Protection

Purpose: Secures sensitive customer and transactional data through encryption.

Implementation Details:

- Creates a **Master Key** encrypted by a password.
- Creates a **Certificate** used to encrypt symmetric keys.
- Creates a **Symmetric Key** to encrypt/decrypt column data such as passwords, emails, and phone numbers.

4. TableLevelCheckConstraints.sql – Advanced Data Validation

Purpose: Adds SQL functions for enforcing data format and validation via check constraints.

Functions Include:

- IsValidEmail() – Validates email format.
- IsValidPhoneNumber() – Validates phone number format.

5. StoredProcedures.sql – Encapsulated Business Logic

Purpose: Implements stored procedures to encapsulate repeatable business operations such as inserting orders and managing stock.

Implemented Stored Procedures:

- InsertShopOrder
    - Inserts a new record into the ShopOrder table.
    - Automatically generates OrderID and OrderDate if not provided.
    - Validates referential integrity: ensures related CustomerID, AddressID, PaymentID, OrderStatusID, and ShippingMethodID exist and are logically connected.
    - Ensures the order date is not in the future and the payment method is not expired.
    - Initializes OrderTotal as 0 (to be calculated later).
    - Returns a success message or detailed validation errors.
- InsertOrderLine
    - Inserts a new record into the OrderLine table.
    - Automatically generates OrderLineID if not provided.
    - Validates OrderID, ProductItemID, and optional CouponID exist.
    - Checks product availability and updates StockQuantity after insertion.
    - Sets initial Price to 0 (expected to be updated by a trigger or calculation function).
    - Returns success or validation error messages.
- DeleteOrderLine
    - Check OrderLine exist.
    - Delete OrderLine
    - Restore the product's stock quantity upon deletion.

6. Trigger.sql – Automated Reactions

Purpose: Implements database triggers to **automate price calculations** and **maintain up-to-date order totals** whenever records in the OrderLine table are inserted, updated, or deleted. These triggers ensure consistent business logic and data integrity across related tables.

Triggers Implemented:

- **trg_OrderLine_CalculatePrice** – Triggered **AFTER INSERT or UPDATE** on OrderLine
    - Calculates OrderLine.Price using the formula: [ProductItem.Price × Quantity × (1 - Coupon.DiscountRate)]
    - If no coupon is applied, a default discount of 0% is used.
    - Updates the associated ShopOrder.OrderTotal by summing all OrderLine.Price values and adding the shipping cost.
- **trg_OrderLine_Delete** – Triggered **AFTER DELETE** from OrderLine
    - Recalculates the ShopOrder.OrderTotal by excluding deleted line items and ensuring the shipping cost is still included.

7. DML.sql – Sample Data and Operations

Purpose: Provides data manipulation scripts to test schema functionality.

8. Views.sql – Simplified Query Interfaces

Purpose: Creates SQL views to abstract and simplify access to frequently queried data, ensuring the following benefits:

- **Enhanced security** (limit access to certain columns).
- **Simplified query logic** (e.g., customer order summary views, product status views).

Design Considerations

1. **Aligns with Business Goals** – Supports key operations such as Customer management, product cataloging, order fulfillment, Coupons, and customer engagement. By directly addressing business needs, the system streamlines operations, enhances shopping experiences, and fosters customer trust. This efficiency not only improves customer satisfaction but also drives higher conversion rates and increases revenue.
2. **Ensures Data Integrity** – We make sure the normalization and referential integrity in our system eliminate data redundancy, ensuring efficient, structured, and consistent data storage. This approach prevents data anomalies, enhances accuracy in transactions, and maintains reliable relationships between entities.
3. **Optimizes Performance** – We enhance system performance by leveraging indexes on primary and foreign keys, improving search and retrieval speed for large datasets, such as finding Customer orders or filtering products by category.
4. **Enhances Security & Compliance** – Secure handling of Customer data, payments, and authentication details prevents unauthorized access. We hash Customer passwords to ensure encrypted storage and protect against security breaches.
5. **Scalable for Future Growth** – The system is built for scalability, supporting multiple currencies, and new payment providers. Its modular design allows easy integration of additional features such as AI-driven recommendations, advanced analytics, and personalized marketing campaigns.

Conclusion

The Target 2.0 E-Commerce Management System is structured to provide an efficient, secure, and scalable solution that meets the needs of both customers and administrators. By aligning entities and relationships with business objectives, the system ensures seamless operations, trust, and digital retail excellence.