ShopStop

Outlet Management System

Database Design Report

CS 432 - Databases Assignment 1

`Project Repository`

Entity-Relationship Diagram

The following ER diagram illustrates the complete database schema for the ShopStop Outlet Management System, showing all entities, their attributes, and the relationships between them.

ER Diagram Components

Entities:

- **Member:** Customer membership information with loyalty program details
- **Employee:** Staff information including hierarchy and shift details
- **Supplier:** Vendor information for product procurement
- **Category:** Product classification with hierarchical structure
- **Product:** Inventory items with pricing and stock details
- **Sale:** Customer transaction records
- **SaleItem:** Individual items within each sale transaction
- **PurchaseOrder:** Orders placed with suppliers
- **OrderItem:** Products included in purchase orders
- **Inventory:** Stock management and location tracking
- **Promotion:** Marketing campaigns and discount programs
- **ProductPromotion:** Association between products and active promotions

ER Diagram Structure

ER Diagram Link: Click here to view the ER Diagram

Note:
In the above ER diagram, entities are represented using rectangles, relationships using diamonds, and attributes using ovals. Primary keys (PK) are indicated by underlining the respective attribute names. Foreign keys (FK) are written in brackets next to the corresponding attribute names.
Although in some ER modeling approaches foreign keys are not explicitly shown since the existence of a relationship itself implies the foreign key they have been included here as per the instructor's requirement for clarity and better alignment with the implemented database schema.
Multiplicity constraints are represented using Crow's Foot (IE) notation instead of numeric labels such as (1:N) or (M:N). These symbols visually convey the cardinality of relationships and are commonly used in practical database design.
Also, participation constraints are not represented in the ER diagram as it is not explicitly specified in the assignment to include, but are discussed in the sections below.
Finally, the ER diagram has been carefully designed to accurately map to the MySQL database schema developed in Module A, ensuring consistency between the conceptual model and its implementation.

The database consists of 12 entities with the following primary keys and key attributes:

Entity Name	Primary Key	Key Attributes
Member	MemberID	Name, Email, MembershipType, LoyaltyPoints
Employee	EmployeeID	Name, Position, Salary, ManagerID (FK)
Supplier	SupplierID	SupplierName, SupplyCategory, Rating
Category	CategoryID	CategoryName, ParentCategoryID (FK)
Product	ProductID	ProductName, Price, StockQuantity, CategoryID (FK)
Sale	SaleID	SaleDate, TotalAmount, MemberID (FK), EmployeeID (FK)
SaleItem	SaleItemID	SaleID (FK), ProductID (FK), Quantity, Subtotal
PurchaseOrder	OrderID	OrderDate, TotalAmount, SupplierID (FK), EmployeeID (FK)
OrderItem	OrderItemID	OrderID (FK), ProductID (FK), QuantityOrdered
Inventory	InventoryID	ProductID (FK), CurrentStock, Location
Promotion	PromotionID	PromotionName, DiscountPercentage, IsActive
ProductPromotion	ProductPromotionID	ProductID (FK), PromotionID (FK)

Relationship Justifications

This section provides detailed justifications for each relationship in the database schema, including cardinality rationale and business logic.

1. **Member - Sale (1:N)**
   Type: One-to-Many (Optional)
   A single member can make multiple purchases over time, but each sale transaction is associated with at most one member. Optional because non-members (walk-in customers) can also make purchases.
   FK: MemberID in Sale table (nullable, ON DELETE SET NULL)

2. **Employee - Sale (1:N)**
   Type: One-to-Many (Mandatory)
   Every sale must be processed by an employee, creating accountability. Each employee can process multiple sales.
   FK: EmployeeID in Sale table (NOT NULL, ON DELETE RESTRICT)

3. **Sale - SaleItem (1:N)**
   Type: One-to-Many (Mandatory)
   Each sale contains one or more product items. Cascade delete ensures that when a sale is removed, all associated items are removed.
   FK: SaleID in SaleItem table (NOT NULL, ON DELETE CASCADE)

4. **Product - SaleItem (1:N)**
   Type: One-to-Many (Mandatory)
   A product can appear in multiple sale transactions. RESTRICT prevents deletion of products with sales history.
   FK: ProductID in SaleItem table (NOT NULL, ON DELETE RESTRICT)

5. **Supplier - Product (1:N)**
   Type: One-to-Many (Mandatory)
   Each supplier provides multiple products, but each product is sourced from a single primary supplier.
   FK: SupplierID in Product table (NOT NULL, ON DELETE RESTRICT)

6. **Category - Product (1:N)**
   Type: One-to-Many (Mandatory)
   Products must belong to exactly one category for organization. Each category can contain multiple products.
   FK: CategoryID in Product table (NOT NULL, ON DELETE RESTRICT)

7. **Category - Category (1:N Self-Reference)**
   Type: One-to-Many (Optional, Recursive)
   Categories can have subcategories. Root categories have NULL parent, subcategories reference their parent.
   FK: ParentCategoryID in Category table (nullable, ON DELETE SET NULL)

8. **Supplier - PurchaseOrder (1:N)**
   Type: One-to-Many (Mandatory)
   Each supplier receives multiple orders, but each order is placed with exactly one supplier.
   FK: SupplierID in PurchaseOrder table (NOT NULL, ON DELETE RESTRICT)

9. **Employee - PurchaseOrder (1:N)**
   Type: One-to-Many (Mandatory)
   Purchase orders must be created by authorized employees. Tracks who initiated procurement.
   FK: EmployeeID in PurchaseOrder table (NOT NULL, ON DELETE RESTRICT)

10. **PurchaseOrder - OrderItem (1:N)**
    Type: One-to-Many (Mandatory)
    Each order contains one or more products. Cascade delete removes items when the parent order is deleted.
    FK: OrderID in OrderItem table (NOT NULL, ON DELETE CASCADE)

11. **Product - OrderItem (1:N)**
    Type: One-to-Many (Mandatory)
    Products can appear in multiple purchase orders. Maintains procurement history analysis.
    FK: ProductID in OrderItem table (NOT NULL, ON DELETE RESTRICT)

12. **Product - Inventory (1:N)**
    Type: One-to-Many (Mandatory)
    Each product have many inventory records. Separates catalog info from dynamic tracking.
    FK: ProductID in Inventory table (NOT NULL, ON DELETE CASCADE)

13. **Product - ProductPromotion (1:N)**
    Type: One-to-Many (via junction table)
    Products can participate in multiple promotions simultaneously. Deletion of product removes links.
    FK: ProductID in ProductPromotion table (NOT NULL, ON DELETE CASCADE)

14. **Promotion - ProductPromotion (1:N)**
    Type: One-to-Many (via junction table)
    Each promotion can apply to multiple products. Deletion of promotion removes product associations.
    FK: PromotionID in ProductPromotion table (NOT NULL, ON DELETE CASCADE)

15. **Employee - Employee (1:N Self-Reference)**
    Type: One-to-Many (Optional, Recursive)
    One manager supervises multiple subordinates. Top-level managers have NULL ManagerID.
    FK: ManagerID in Employee table (nullable, ON DELETE SET NULL)

Participation Constraints and Weak Entities

Participation Constraints

Participation constraints define whether an entity's existence is dependent (total participation) or optional (partial participation). In this schema, these are enforced via `NOT NULL` foreign keys and referential actions.

Total Participation (Mandatory)
The following entities **cannot exist** without their parent entity. These use `NOT NULL` constraints to prevent orphan records:

- Product → Category
- Product → Supplier
- Sale → Employee
- SaleItem → Sale
- SaleItem → Product
- PurchaseOrder → Supplier
- PurchaseOrder → Employee
- OrderItem → PurchaseOrder
- OrderItem → Product
- Inventory → Product
- ProductPromotion → Product
- ProductPromotion → Promotion

Partial Participation (Optional)
Partial participation allows an entity to exist independently of a specific relationship, implemented using nullable foreign keys:

- **Category self-relationship**: Root categories have a nullable `ParentCategoryID`.
- **Sale** → **Member**: `MemberID` is nullable to support guest/walk-in customers.
- **Employee self-relationship**: Top-level managers have a nullable `ManagerID`.
- **PurchaseOrder Delivery**: `ActualDeliveryDate` remains null for pending orders.

Weak Entities

Weak entities depend on parent entities for identification and existence. In this schema, they are enforced using `ON DELETE CASCADE` to ensure they are removed if the owner entity is deleted.

Weak Entity	Owner Entity (Parent)
SaleItem	Sale
OrderItem	PurchaseOrder
ProductPromotion	Product and Promotion
Inventory	Product

Strong Entities: All other entities (*Member, Employee, Supplier, Category, Product, Sale, PurchaseOrder, Promotion*) are strong entities as they possess their own independent primary keys and can exist standalone.

Additional Constraints

1. Domain Constraints

Table-Level Rules

Member Table: Age $\geq$ 18; LoyaltyPoints $\geq$ 0; MembershipType: Silver, Gold, Platinum; Unique Email.
Employee Table: Salary $>$ 0; Unique Email; No circular ManagerID references.
Supplier Table: Rating between 0.00 and 5.00; Unique Email.
Product Table: Price $>$ 0; StockQuantity $\geq$ 0; ReorderLevel $\geq$ 0; Unique Barcode; ExpiryDate $>$ ManufactureDate.
Sale Table: Non-negative amounts; FinalAmount = TotalAmount - DiscountAmount; PaymentMethod: Cash, Card, UPI, Wallet.
Promotion Table: DiscountPercentage (1–100); EndDate $\geq$ StartDate.

2. Tuple-Based Constraints

SaleItem: Subtotal = Quantity $\times$ UnitPrice; Quantity > 0; UnitPrice > 0.
OrderItem: Subtotal = QuantityOrdered $\times$ UnitCost; QuantityOrdered > 0; UnitCost > 0.
PurchaseOrder: ExpectedDeliveryDate $\geq$ OrderDate; Status: Pending, Confirmed, Delivered, Cancelled.
Inventory: CurrentStock ≥ 0; LastRestockQuantity > 0.

3. Uniqueness & NOT NULL Constraints

Unique constraints apply to Emails (Member, Employee, Supplier), CategoryName, and Product Barcodes. All tables contain at least three mandatory (NOT NULL) columns beyond the PK, ensuring business-critical data like HireDate, SaleDate, and ProductPrice are always captured.

4. Referential Integrity Actions

- **ON DELETE CASCADE:** Sale $\rightarrow$ SaleItem, PurchaseOrder $\rightarrow$ OrderItem, Product $\rightarrow$ Inventory, Product/Promotion $\rightarrow$ ProductPromotion.
- **ON DELETE RESTRICT:** Product $\rightarrow$ SaleItem/OrderItem, Category $\rightarrow$ Product, Supplier $\rightarrow$ Product/PurchaseOrder, Employee $\rightarrow$ Sale/PurchaseOrder.
- **ON DELETE SET NULL:** Member $\rightarrow$ Sale, ManagerID $\rightarrow$ Employee, ParentCategoryID $\rightarrow$ Category.

5. Business Logic Constraints

The system enforces automated logic to ensure operational efficiency. Restocking alerts are triggered whenever **StockQuantity** falls below the **ReorderLevel**, while **Inventory CurrentStock** is strictly synchronized with the **Product StockQuantity**. Sales to registered members automatically accumulate **LoyaltyPoints** based on the **FinalAmount**, and tiered discounts are applied exclusively to members according to their **MembershipType** (Silver, Gold, or Platinum). Additionally, products approaching their **ExpiryDate** are flagged for clearance, **Promotions** remain active only within their defined **StartDate** and **EndDate** range, and **PurchaseOrder** statuses must progress through a logical workflow (Pending $\rightarrow$ Confirmed $\rightarrow$ Delivered).

Team Member Contributions

Chepuri Venkata Naga Thrisha	**(23110079)**	Contributed to report content preparation and documentation formatting.
Katta Revathi	**(23110159)**	Designed and finalized the ER diagram based on the implemented schema.
Pappala Sai Keerthana	**(23110229)**	Led the report writing and coordinated overall documentation and structure.
Ravi Bhavana	**(23110274)**	Assisted in database schema implementation and coding tasks.
Thoutam Dhruthika	**(23110340)**	Worked on SQL implementation and contributed to parts of the report.