Term Project: E-commerce Website Database System
CS669 Database Design and Implementation for Business

Author: Natasha Machado

BOSTON UNIVERSITY Metropolitan College

Fall 2023

Contents

Overview

In this project, we will design and implement a relational database system for an e-commerce website. For this project, we will stick to five primary aspects which are new products, product delivery, new customer accounts, product purchases, and product shipment.

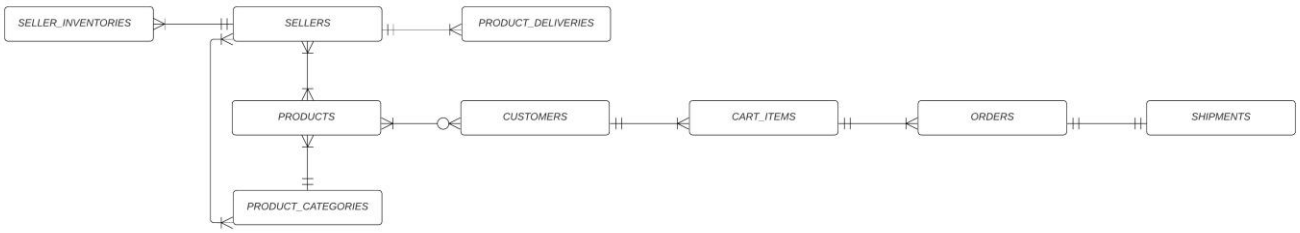
Structural Database Rules

- If a customer/seller has or creates an account, they have placed/sold at least one order/product.
- Each product category at least has one product, or the category does not exist until a product listing is created.
- All items in the customers' cart are sent to the warehouse by the same or different sellers and the customer orders are then combined and sent to the customer from the warehouse.
- Company tracks seller inventory stocks, product deliveries, and shipments and keeps the same updated.

Relationships

- Each seller sells one or more products in one or more categories and each product and category is sold by at least one or more sellers.
- Each product category can have multiple products, but each product can belong to only one product category.
- Each seller can have one or more inventories, but each inventory should belong to only one seller.
- Each seller can send multiple product deliveries, but each product delivery should come from only one seller.
- Each customer can buy one or more products and each product can be bought by many or no customers.
- Each customer can have multiple cart items, but each cart item should be linked to one customer.
- One or many orders can be sent from one cart, but each order should come from one cart only.
- One order should be linked to only one shipment.

Conceptual Entity Relationship Diagram (ERD)



MS SQL Server-Based Constraints and Datatypes

Entity: **SELLERS**
Attributes:

seller_id (INT, NOT NULL, IDENTITY, PRIMARY KEY)
seller_name (VARCHAR(255), NOT NULL)
seller_email (VARCHAR(255), NOT NULL)
seller_password (VARCHAR(255), NOT NULL)
seller_address (VARCHAR(255), NOT NULL)
seller_phone (VARCHAR(255), NOT NULL)

Entity: **CUSTOMERS**
Attributes:

customer_id (INT, NOT NULL, IDENTITY, PRIMARY KEY)
customer_name (VARCHAR(255), NOT NULL)
customer_email (VARCHAR(255), NOT NULL)
customer_password (VARCHAR(255), NOT NULL)
customer_address (VARCHAR(255), NOT NULL)
customer_phone (VARCHAR(255), NOT NULL)

Entity: **PRODUCT_CATEGORIES**
Attributes:

category_id (INT, NOT NULL, IDENTITY, PRIMARY KEY)
product_category_name (VARCHAR(255), NOT NULL)

Entity: **PRODUCTS**
Attributes:

product_id (INT, NOT NULL, IDENTITY, PRIMARY KEY)
product_name (VARCHAR(255), NOT NULL)
product_description (VARCHAR(255), NOT NULL)
product_price (DECIMAL(18,2), NOT NULL)
category_id (INT, NOT NULL, FOREIGN KEY REFERENCES PRODUCT_CATEGORIES(category_id))

Entity: **SELLER_INVENTORIES**
Attributes:
- seller_inventory_id (INT, NOT NULL, IDENTITY, PRIMARY KEY)
- product_stock (INT, NOT NULL)
- seller_id (INT, NOT NULL, FOREIGN KEY REFERENCES SELLERS(seller_id))
- product_id (INT, NOT NULL, FOREIGN KEY REFERENCES PRODUCTS(product_id))

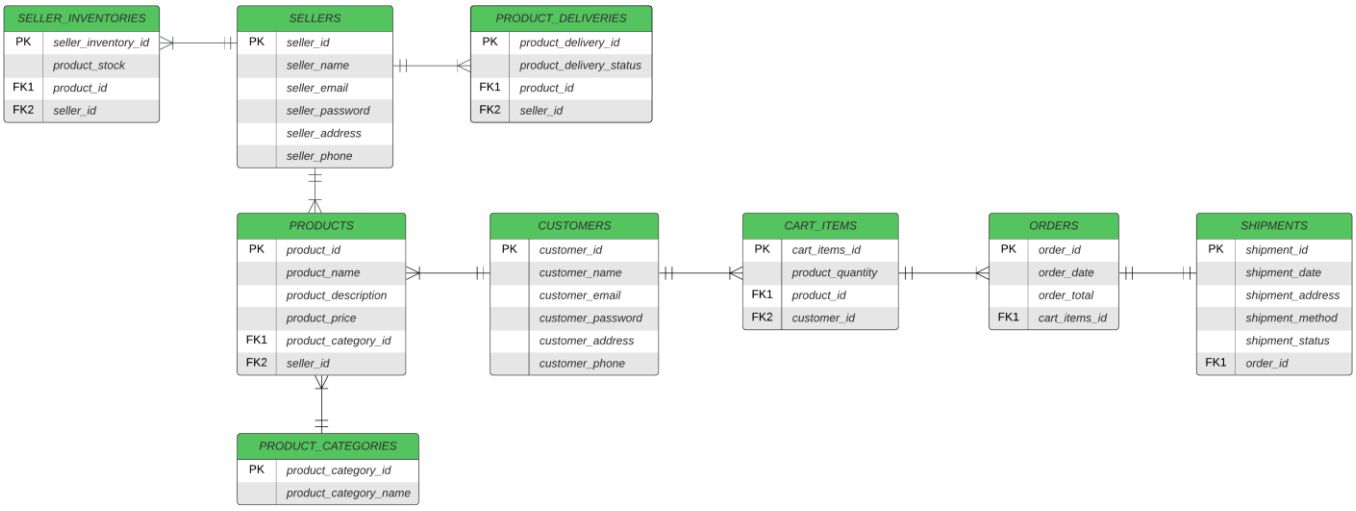
Entity: **PRODUCT_DELIVERIES**
Attributes:
- product_delivery_id (INT, NOT NULL, IDENTITY, PRIMARY KEY)
- product_delivery_status (VARCHAR(255), NOT NULL)
- product_id (INT, NOT NULL, FOREIGN KEY REFERENCES PRODUCTS(product_id))
- seller_id (INT, NOT NULL, FOREIGN KEY REFERENCES SELLERS(seller_id))

Entity: **CART_ITEMS**
Attributes:
- cart_item_id (INT, NOT NULL, IDENTITY, PRIMARY KEY)
- product_quantity (INT, NOT NULL)
- product_id (INT, NOT NULL, FOREIGN KEY REFERENCES PRODUCTS(product_id))
- customer_id (INT, NOT NULL, FOREIGN KEY REFERENCES CUSTOMERS(customer_id))

Entity: **ORDERS**
Attributes:
- order_id (INT, NOT NULL, IDENTITY, PRIMARY KEY)
- order_total (DECIMAL(18,2), NOT NULL)
- order_date (DATETIME, NOT NULL)
- cart_item_id (INT, NOT NULL, FOREIGN KEY REFERENCES CART_ITEMS(cart_item_id))

Entity: **SHIPMENTS**
Attributes:
- shipment_id (INT, NOT NULL, IDENTITY, PRIMARY KEY)
- shipment_date (DATETIME, NOT NULL)
- shipment_address (VARCHAR(255), NOT NULL)
- shipment_method (VARCHAR(255), NOT NULL)
- shipment_status (VARCHAR(255), NOT NULL)
- order_id (INT, NOT NULL, FOREIGN KEY REFERENCES ORDERS(order_id))

Initial Logical Entity Relationship Diagram (ERD)



Tables, sequences, and constraints in SQL

```sql
CREATE TABLE SELLERS (
    seller_id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
    seller_name VARCHAR(255) NOT NULL,
    seller_email VARCHAR(255) NOT NULL,
    seller_password VARCHAR(255) NOT NULL,
    seller_address VARCHAR(255) NOT NULL,
    seller_phone VARCHAR(255) NOT NULL
);

CREATE TABLE CUSTOMERS (
    customer_id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
    customer_name VARCHAR(255) NOT NULL,
    customer_email VARCHAR(255) NOT NULL,
    customer_password VARCHAR(255) NOT NULL,
    customer_address VARCHAR(255) NOT NULL,
    customer_phone VARCHAR(255) NOT NULL
);

CREATE TABLE PRODUCT_CATEGORIES (
    category_id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
    product_category_name VARCHAR(255) NOT NULL
);

CREATE TABLE PRODUCTS (
    product_id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
    product_name VARCHAR(255) NOT NULL,
    product_description VARCHAR(255) NOT NULL,
    product_price DECIMAL(18, 2) NOT NULL,
    category_id INT NOT NULL,
    CONSTRAINT FK_Category FOREIGN KEY (category_id) REFERENCES PRODUCT_CATEGORIES(category_id)
);
```

```sql
CREATE TABLE SELLER_INVENTORIES (
    seller_inventory_id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
    product_stock INT NOT NULL,
    seller_id INT NOT NULL,
    product_id INT NOT NULL,
    CONSTRAINT FK_Seller FOREIGN KEY (seller_id) REFERENCES SELLERS(seller_id),
    CONSTRAINT FK_Product FOREIGN KEY (product_id) REFERENCES PRODUCTS(product_id)
);

CREATE TABLE PRODUCT_DELIVERIES (
    product_delivery_id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
    product_delivery_status VARCHAR(255) NOT NULL,
    product_id INT NOT NULL,
    seller_id INT NOT NULL,
    CONSTRAINT FK_ProductDelivery_Product FOREIGN KEY (product_id) REFERENCES PRODUCTS(product_id),
    CONSTRAINT FK_ProductDelivery_Seller FOREIGN KEY (seller_id) REFERENCES SELLERS(seller_id)
);

CREATE TABLE CART_ITEMS (
    cart_item_id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
    product_quantity INT NOT NULL,
    product_id INT NOT NULL,
    customer_id INT NOT NULL,
    CONSTRAINT FK_CartItem_Product FOREIGN KEY (product_id) REFERENCES PRODUCTS(product_id),
    CONSTRAINT FK_CartItem_Customer FOREIGN KEY (customer_id) REFERENCES CUSTOMERS(customer_id)
);

CREATE TABLE ORDERS (
    order_id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
    order_total DECIMAL(18,2) NOT NULL,
    order_date DATETIME NOT NULL,
    cart_item_id INT NOT NULL,
    CONSTRAINT FK_Order_CartItem FOREIGN KEY (cart_item_id) REFERENCES CART_ITEMS(cart_item_id)
);
```

```sql
CREATE TABLE SHIPMENTS (
    shipment_id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
    shipment_date DATETIME NOT NULL,
    shipment_address VARCHAR(255) NOT NULL,
    shipment_method VARCHAR(255) NOT NULL,
    shipment_status VARCHAR(255) NOT NULL,
    order_id INT NOT NULL,
    CONSTRAINT FK_Shipment_Order FOREIGN KEY (order_id) REFERENCES ORDERS(order_id)
);
```

Use Case-Driven Aspects

Aspect 1: New Products

```sql
INSERT INTO SELLERS (seller_name, seller_email, seller_password, seller_address, seller_phone)
VALUES
    ('Taylor Swift', 'taylorswift@example.com', 'taylorswift@13', 'Cornelia St', '526-7854'),
    ('Selena Gomez', 'selenagomez@example.com', 'selenagomez@98', 'Rare St', '951-2579');

INSERT INTO CUSTOMERS (customer_name, customer_email, customer_password, customer_address, customer_phone)
VALUES
    ('Sheldon Cooper', 'sheldoncooper@bigbang.com', 'Sheldon@34', '46 Pasadena', '575-3953'),
    ('Penny', 'penny@bigbang.com', 'penny', '45 Pasadena', '595-1594');

INSERT INTO PRODUCT_CATEGORIES (product_category_name) VALUES ('Computers');
INSERT INTO PRODUCT_CATEGORIES (product_category_name) VALUES ('Electronics');

INSERT INTO PRODUCTS (product_name, product_description, product_price, category_id)
VALUES
    ('Laptop', 'High-performance laptop', 899.99, 3),
    ('Smartphone', 'Latest smartphone model', 599.99, 4);

INSERT INTO SELLER_INVENTORIES (product_stock, seller_id, product_id)
VALUES
    (50, 1, 39),
    (30, 2, 40);
```

```sql
-- New Products
CREATE PROCEDURE AddNewProduct (
    @p_Name VARCHAR(255),
    @p_Description VARCHAR(255),
    @p_Price DECIMAL(18, 2),
    @p_CategoryID INT
)
AS
BEGIN
    INSERT INTO PRODUCTS (product_name, product_description, product_price, category_id)
    VALUES (@p_Name, @p_Description, @p_Price, @p_CategoryID);
END;

-- Invoke stored procedure to add new products
EXEC AddNewProduct 'Floating Lamp', 'Floats in air', 13.99, 3;
EXEC AddNewProduct 'Throwable Camera', 'Capture photos in flight', 49.99, 4;

-- Query for products that cost $25 or less
SELECT *
FROM PRODUCTS
WHERE category_id IN (3, 4) AND product_price <= 25;
```

	product_id	product_name	product_description	product_price	category_id
1	45	Floating Lamp	Floats in air	13.99	3

Aspect 2: Product Delivery

```sql
-- Product Delivery
CREATE PROCEDURE ProductDelivery (
    @p_ProductID INT,
    @p_SellerID INT,
    @p_UnitsDelivered INT,
    @p_Condition VARCHAR(255)
)
AS
BEGIN
    UPDATE SELLER_INVENTORIES
    SET product_stock = product_stock + @p_UnitsDelivered
    WHERE product_id = @p_ProductID AND seller_id = @p_SellerID;

    INSERT INTO PRODUCT_DELIVERIES (product_delivery_status, product_id, seller_id)
    VALUES (@p_Condition, @p_ProductID, @p_SellerID);
END;

-- Invoke stored procedure to update inventory for the new products
EXEC ProductDelivery 39, 1, 10, 'New';
EXEC ProductDelivery 40, 2, 15, 'New';

-- Query for seller's products with an inventory of 15 or more
SELECT S.seller_name, P.product_name, SI.product_stock
FROM SELLERS S
JOIN SELLER_INVENTORIES SI ON S.seller_id = SI.seller_id
JOIN PRODUCTS P ON SI.product_id = P.product_id
WHERE SI.product_stock >= 15;
```

	seller_name	product_name	product_stock
1	Taylor Swift	Laptop	60
2	Selena Gomez	Smartphone	45

Aspect 3: New Customer Accounts

```sql
--New Customer Accounts
CREATE PROCEDURE AddNewCustomer (
    @p_Name VARCHAR(255),
    @p_Email VARCHAR(255),
    @p_Password VARCHAR(255),
    @p_Address VARCHAR(255),
    @p_Phone VARCHAR(255)
)
AS
BEGIN
    INSERT INTO CUSTOMERS (customer_name, customer_email, customer_password, customer_address, customer_phone)
    VALUES (@p_Name, @p_Email, @p_Password, @p_Address, @p_Phone);
END;

--Invoke stored procedure to add new customers
EXEC AddNewCustomer 'Carter Channing', 'carter@gmail.com', 'password123', '456 Oak St', '555-5678';
EXEC AddNewCustomer 'Delaney Emerson', 'delaney@gmail.com', 'password456', '789 Pine St', '555-6789';

--Query for last names of customers with at least 3 accounts and the number of accounts
SELECT customer_name, COUNT(*) AS num_accounts
FROM CUSTOMERS
GROUP BY customer_name
HAVING COUNT(*) >= 3;
```

```sql
SELECT customer_name, COUNT(*) AS num_accounts
FROM CUSTOMERS
GROUP BY customer_name
HAVING COUNT(*) >= 1;
```

100 %

Results | Messages

	customer_name	num_accounts
1	Carter Channing	1
2	Delaney Emerson	1
3	Penny	1
4	Sheldon Cooper	1

```sql
SELECT customer_name, COUNT(*) AS num_accounts
FROM CUSTOMERS
GROUP BY customer_name
HAVING COUNT(*) >= 3;
```

100 %

Results | Messages

customer_name	num_accounts

```sql
--Create the PRODUCTS_HISTORY table
CREATE TABLE PRODUCTS_HISTORY (
    product_id INT NOT NULL,
    product_name VARCHAR(255) NOT NULL,
    product_description VARCHAR(255) NOT NULL,
    product_price DECIMAL(18, 2) NOT NULL,
    category_id INT NOT NULL,
    change_date DATETIME NOT NULL
);

--Create the trigger to update the history table
CREATE TRIGGER trg_Products_History
ON PRODUCTS
AFTER UPDATE
AS
BEGIN
    INSERT INTO PRODUCTS_HISTORY (product_id, product_name, product_description, product_price, category_id, change_date)
    SELECT
        COALESCE(INSERTED.product_id, DELETED.product_id),
        COALESCE(INSERTED.product_name, DELETED.product_name),
        COALESCE(INSERTED.product_description, DELETED.product_description),
        COALESCE(INSERTED.product_price, DELETED.product_price),
        COALESCE(INSERTED.category_id, DELETED.category_id),
        GETDATE()
    FROM INSERTED
    INNER JOIN DELETED ON INSERTED.product_id = DELETED.product_id;
END;
```

Aspect 4: Product Purchases

```sql
--Product Purchases
CREATE PROCEDURE ProductPurchase (
    @p_CustomerID INT,
    @p_ProductID INT,
    @p_Quantity INT
)
AS
BEGIN
    DECLARE @OrderTotal DECIMAL(18, 2);

    -- Calculate the order total based on product price and quantity
    SELECT @OrderTotal = product_price * @p_Quantity
    FROM PRODUCTS
    WHERE product_id = @p_ProductID;

    -- Insert into CART_ITEMS
    INSERT INTO CART_ITEMS (product_quantity, product_id, customer_id)
    VALUES (@p_Quantity, @p_ProductID, @p_CustomerID);

    -- Insert into ORDERS
    INSERT INTO ORDERS (order_total, order_date, cart_item_id)
    VALUES (@OrderTotal, GETDATE(), SCOPE_IDENTITY());
END;

--Invoke stored procedure to make product purchases
EXEC ProductPurchase 1, 39, 2;
EXEC ProductPurchase 2, 40, 1;
```

--Query for names and addresses of customers who bought products purchased by at least four people

```sql
SELECT DISTINCT C.customer_name, C.customer_address
FROM CUSTOMERS C
JOIN CART_ITEMS CI ON C.customer_id = CI.customer_id
JOIN ORDERS O ON CI.cart_item_id = O.cart_item_id
JOIN CART_ITEMS CI2 ON O.cart_item_id = CI2.cart_item_id
GROUP BY C.customer_name, C.customer_address
HAVING COUNT(DISTINCT CI2.customer_id) >= 4;
```

100 %

Results Messages

customer_name	customer_address

```sql
SELECT DISTINCT C.customer_name, C.customer_address
FROM CUSTOMERS C
JOIN CART_ITEMS CI ON C.customer_id = CI.customer_id
JOIN ORDERS O ON CI.cart_item_id = O.cart_item_id
JOIN CART_ITEMS CI2 ON O.cart_item_id = CI2.cart_item_id
GROUP BY C.customer_name, C.customer_address
HAVING COUNT(DISTINCT CI2.customer_id) >= 1;
```

100 %

Results Messages

	customer_name	customer_address
1	Penny	45 Pasadena
2	Sheldon Cooper	46 Pasadena

Aspect 5: Product Shipment

```sql
--Product Shipment
CREATE PROCEDURE ProductShipment (
    @p_OrderID INT,
    @p_ShipmentAddress VARCHAR(255),
    @p_ShipmentMethod VARCHAR(255)
)
AS
BEGIN
    -- Insert into SHIPMENTS
    INSERT INTO SHIPMENTS (shipment_date, shipment_address, shipment_method, shipment_status, order_id)
    VALUES (GETDATE(), @p_ShipmentAddress, @p_ShipmentMethod, 'Shipped', @p_OrderID);
END;

--Invoke stored procedure to ship the orders from Aspect 4
EXEC ProductShipment 1, '123 Oak St', 'Standard Shipping';
EXEC ProductShipment 2, '456 Pine St', 'Two-Day Shipping';
```

```sql
--Define and execute a custom query for this aspect, let's retrieve the shipment information for orders shipped using 'Standard Shipping'
SELECT S.shipment_id, S.shipment_date, S.shipment_address, S.shipment_method
FROM SHIPMENTS S
JOIN ORDERS O ON S.order_id = O.order_id
WHERE S.shipment_method = 'Standard Shipping';
```

100 %

Results Messages

	shipment_id	shipment_date	shipment_address	shipment_method
1	1	2023-11-28 12:26:16.877	123 Oak St	Standard Shipping