Database Design Project

Siavash Pourfallah
This project is meant for the database design course in the Computer Science Department of Amirkabir University of Technology.

Phase 1: ER Diagram Design

ER Diagram Relations

A description of the relations and tables we are dealing with includes:

1. Users

- **Primary Key:** `user_id`
- **Columns:** `user_id`, `username`, `password`, `name`, `email`, `contact_number`, `address`, `registration_date`
- **Description:** Stores information about users registered on the platform, including login credentials and contact details.
- **Cardinalities:**
  - `User` to `Orders`: 1:N
  - `User` to `ShoppingCart`: 1:N
  - `User` to `PurchaseHistory`: 1:N
  - `User` to `Comments`: 1:N

2. Managers

- **Primary Key:** `manager_id`
- **Columns:** `manager_id`, `username`, `password`, `email`, `registration_date`
- **Description:** Stores information about managers who moderate comments.
- **Cardinalities:**
  - `Manager` to `Comments`: 1:N

3. Categories

- **Primary Key:** `category_id`
- **Columns:** `category_id`, `name`
- **Description:** Stores different categories for products.
- **Cardinalities:**
  - `Category` to `Products`: 1:N

4. Brands

- **Primary Key:** `brand_id`
- **Columns:** `brand_id`, `name`, `status`
- **Description:** Stores information about brands associated with products. The `status` column can be `active`, `inactive` or `old`.

- **Cardinalities:**
  - `Brand` to `Products`: 1:N

5. Products

- **Primary Key:** `product_id`
- **Columns:** `product_id`, `name`, `description`, `price`, `stock`, `category_id`, `brand_id`, `created_at`, `status`
- **Description:** Stores detailed information about products available on the platform. The `status` column can be
- **Cardinalities:**
  - `Product` to `OrderDetails`: 1:N
  - `Product` to `CartItems`: 1:N
  - `Product` to `Comments`: 1:N
  - `Product` to `ProductDiscounts`: 1:N
  - `Category` to `Product`: N:1
  - `Brand` to `Product`: N:1

6. ShippingInfo

- **Primary Key:** `shipping_info_id`
- **Columns:** `shipping_info_id`, `tracking_number`, `carrier`, `shipping_date`, `delivery_date`, `status`
- **Description:** Stores shipping details for orders, including tracking and delivery information. The `status` column can be `pending`, `shipped`, `delivered` or `returned`.
- **Cardinalities:**
  - `ShippingInfo` to `Orders`: 1:N

7. Orders

- **Primary Key:** `order_id`
- **Columns:** `order_id`, `user_id`, `order_date`, `status`, `total_amount`, `shipping_info_id`
- **Description:** Stores information about orders placed by users. Status can be `pending`, `processing`, `shipped`, `delivered`, `canceled` or `returned`.
- **Cardinalities:**
  - `Order` to `OrderDetails`: 1:N
  - `Order` to `PurchaseHistory`: 1:N
  - `User` to `Order`: N:1

    - `ShippingInfo` to `Order`: N:1

8. OrderDetails

- **Primary Key:** `order_detail_id`
- **Columns:** `order_detail_id`, `order_id`, `product_id`, `quantity`, `price`
- **Description:** Stores details of each product in an order, including quantity and price.
- **Cardinalities:**
    - `OrderDetail` to `Order`: N:1
    - `OrderDetail` to `Product`: N:1

9. ShoppingCart

- **Primary Key:** `cart_id`
- **Columns:** `cart_id`, `user_id`, `created_at`, `updated_at`
- **Description:** Stores shopping cart information for users.
- **Cardinalities:**
    - `ShoppingCart` to `CartItems`: 1:N
    - `User` to `ShoppingCart`: 1:N

10. CartItems

- **Primary Key:** `cart_item_id`
- **Columns:** `cart_item_id`, `cart_id`, `product_id`, `quantity`
- **Description:** Stores individual items in a user's shopping cart.
- **Cardinalities:**
    - `CartItem` to `ShoppingCart`: N:1
    - `CartItem` to `Product`: N:1

11. PurchaseHistory

- **Primary Key:** `history_id`
- **Columns:** `history_id`, `user_id`, `order_id`, `purchase_date`
- **Description:** Stores historical purchase information for users.
- **Cardinalities:**
    - `PurchaseHistory` to `User`: N:1
    - `PurchaseHistory` to `Order`: N:1

12. Comments

- **Primary Key:** `comment_id`

- **Columns:** `comment_id`, `product_id`, `user_id`, `comment`, `comment_date`, `status`, `moderated_by`
- **Description:** Stores user comments on products and their moderation status. The `status` column can be `approved` or `inappropriate`.
- **Cardinalities:**
    - `Comment` to `Product`: N:1
    - `Comment` to `User`: N:1
    - `Comment` to `Manager`: N:1 (each comment is managed by a single manager)

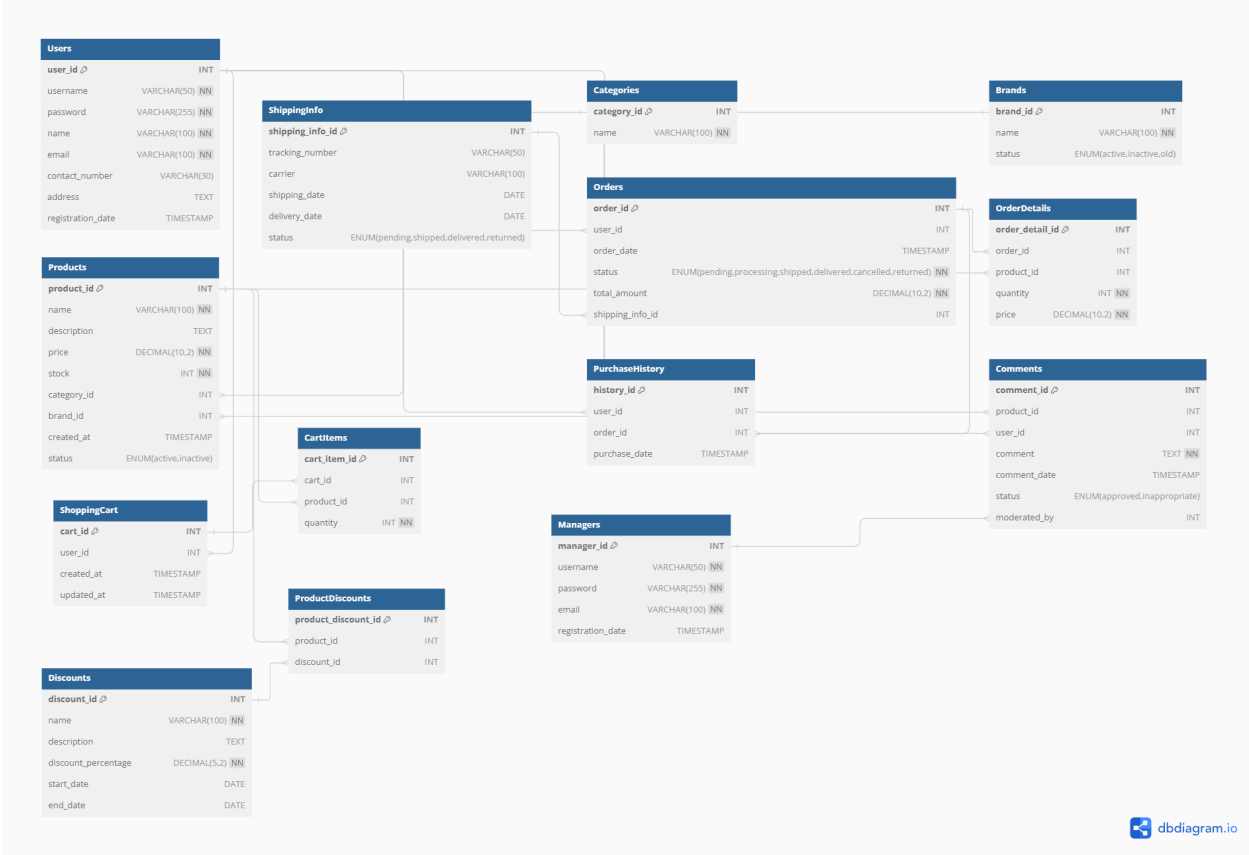
13. Discounts

- **Primary Key:** `discount_id`
- **Columns:** `discount_id`, `name`, `description`, `discount_percentage`, `start_date`, `end_date`
- **Description:** Stores information about discounts and special offers.
- **Cardinalities:**
    - `Discount` to `ProductDiscounts`: 1:N

14. ProductDiscounts

- **Primary Key:** `product_discount_id`
- **Columns:** `product_discount_id`, `product_id`, `discount_id`
- **Description:** Stores the association between products and discounts.
- **Cardinalities:**
    - `ProductDiscount` to `Product`: N:1
    - `ProductDiscount` to `Discount`: N:1

Final ER Diagram



[View this ER diagram on dbdiagram.io]

Phase 2: Normalization

Normalization Process

1. Users

- **1NF**: All attributes contain atomic values, and each attribute holds a single value.
- **2NF**: The primary key is `user_id`. All non-key attributes (username, password, name, email, contact_number, address, registration_date) are fully functionally dependent on `user_id`.
- **3NF**: There are no transitive dependencies. Every non-key attribute is directly dependent on the primary key.

2. Managers

- **1NF**: All attributes contain atomic values, and each attribute holds a single value.
- **2NF**: The primary key is `manager_id`. All non-key attributes (username, password, email, registration_date) are fully functionally dependent on `manager_id`.
- **3NF**: There are no transitive dependencies. Every non-key attribute is directly dependent on the primary key.

3. Categories

- **1NF**: All attributes contain atomic values, and each attribute holds a single value.
- **2NF**: The primary key is `category_id`. The `name` attribute is fully functionally dependent on `category_id`.
- **3NF**: There are no transitive dependencies.

4. Brands

- **1NF**: All attributes contain atomic values, and each attribute holds a single value.
- **2NF**: The primary key is `brand_id`. All non-key attributes (name, status) are fully functionally dependent on `brand_id`.
- **3NF**: There are no transitive dependencies.

5. Products

- **1NF**: All attributes contain atomic values, and each attribute holds a single value.
- **2NF**: The primary key is `product_id`. All non-key attributes (name, description, price, stock, category_id, brand_id, created_at, status) are fully functionally dependent on `product_id`.
- **3NF**: There are no transitive dependencies.

6. ShoppingInfo

- **1NF**: All attributes contain atomic values, and each attribute holds a single value.
- **2NF**: The primary key is `shipping_info_id`. All non-key attributes (tracking_number, carrier, shipping_date, delivery_date, status) are fully functionally dependent on `shipping_info_id`.
- **3NF**: There are no transitive dependencies.

7. Orders

- **1NF**: All attributes contain atomic values, and each attribute holds a single value.
- **2NF**: The primary key is `order_id`. All non-key attributes (user_id, order_date, status, total_amount, shipping_info_id) are fully functionally dependent on `order_id`.
- **3NF**: There are no transitive dependencies.

8. OrderDetails

- **1NF**: All attributes contain atomic values, and each attribute holds a single value.
- **2NF**: The primary key is `order_detail_id`. All non-key attributes (order_id, product_id, quantity, price) are fully functionally dependent on `order_detail_id`.
- **3NF**: There are no transitive dependencies.

9. ShoppingCart

- **1NF**: All attributes contain atomic values, and each attribute holds a single value.
- **2NF**: The primary key is `cart_id`. All non-key attributes (user_id, created_at, updated_at) are fully functionally dependent on `cart_id`.
- **3NF**: There are no transitive dependencies.

10. CartItems

- **1NF**: All attributes contain atomic values, and each attribute holds a single value.
- **2NF**: The primary key is `cart_item_id`. All non-key attributes (cart_id, product_id, quantity) are fully functionally dependent on `cart_item_id`.
- **3NF**: There are no transitive dependencies.

11. PurchaseHistory

- **1NF**: All attributes contain atomic values, and each attribute holds a single value.
- **2NF**: The primary key is `history_id`. All non-key attributes (user_id, order_id, purchase_date) are fully functionally dependent on `history_id`.
- **3NF**: There are no transitive dependencies.

12. Comments

- **1NF**: All attributes contain atomic values, and each attribute holds a single value.
- **2NF**: The primary key is `comment_id`. All non-key attributes (product_id, user_id, comment, comment_date, status, moderated_by) are fully functionally dependent on `comment_id`.
- **3NF**: There are no transitive dependencies.

13. Discounts

- **1NF**: All attributes contain atomic values, and each attribute holds a single value.
- **2NF**: The primary key is `discount_id`. All non-key attributes (name, description, discount_percentage, start_date, end_date) are fully functionally dependent on `discount_id`.
- **3NF**: There are no transitive dependencies.

14. ProductDiscounts

- **1NF**: All attributes contain atomic values, and each attribute holds a single value.
- **2NF**: The primary key is `product_discount_id`. All non-key attributes (product_id, discount_id) are fully functionally dependent on `product_discount_id`.
- **3NF**: There are no transitive dependencies.

Phase 3: Creating Tables and Adding Data

The SQL script defines all the tables in the `onlineshop` database in the `onlineshop.sql` file.

The provided Python script automates the process of populating an online shopping database with synthetic data using the `Faker` library and `mysql-connector-python`. The fake data is in the `fake_data.py` file.

Phase 4: Queries Using Python

List all inactive brands and the number of their products.

```
SELECT b.name AS brand , COUNT(p.product_id) AS product_count
FROM Brands b
JOIN Products p ON b.brand_id = p.brand_id
WHERE b.status = 'inactive'
GROUP BY b.name;
```

Retrieve all users who registered in the last specified number of days.

```
SELECT username, email, registration_date
FROM Users
WHERE registration_date >= DATE_SUB(CURDATE(), INTERVAL %s DAY);
```

Get all products along with their category names.

```
SELECT p.name AS product_name, c.name AS category_name
FROM Products p
JOIN Categories c ON p.category_id = c.category_id;
```

Find all orders placed by a specific user (given user_id) and their total amounts.

```
SELECT o.order_id, o.order_date, o.total_amount
FROM Orders o
WHERE o.user_id = %s;
```

Get the details of products that have more than the specified number of units in stock.

```
SELECT name, description, price, stock
FROM Products
WHERE stock > %s;
```

List the details of all orders and their shipping information.

```
SELECT o.order_id, o.order_date, o.status, s.tracking_number, s.carrier,
s.shipping_date, s.delivery_date
```

```
FROM Orders o
JOIN ShippingInfo s ON o.shipping_info_id = s.shipping_info_id;
```

Find all users who have made a purchase and the total amount they have spent.

```
SELECT u.user_id, u.username, SUM(o.total_amount) AS total_spent
FROM Users u
JOIN Orders o ON u.user_id = o.user_id
GROUP BY u.user_id, u.username;
```

Retrieve all approved comments for a specific product (given product_id).

```
SELECT c.comment, c.comment_date, u.username
FROM Comments c
JOIN Users u ON c.user_id = u.user_id
WHERE c.product_id = %s AND c.status = 'approved';
```

Get all products along with their discounts, if any.

```
SELECT p.name AS product_name, d.name AS discount_name,
d.discount_percentage
FROM Products p
LEFT JOIN ProductDiscounts pd ON p.product_id = pd.product_id
LEFT JOIN Discounts d ON pd.discount_id = d.discount_id;
```

Retrieve the total number of products in each category.

```
SELECT c.name AS category_name, COUNT(p.product_id) AS product_count
FROM Categories c
JOIN Products p ON c.category_id = p.category_id
GROUP BY c.name;
```

These queries have been implemented in the `queries.py` file.