Digital Egypt Pioneers

DATABASE DESIGN ANALYSIS

OpenCart Demo –
Online Shopping Platform

Submitted by:

Nervana Issac
Hadeer Mahmoud
Amira Samy
Youssef Khaled

Digital Egypt Pioneers

1. Objective

To understand the core database structure of the OpenCart platform (simulated via application behavior) and design test cases to verify data integrity and consistency following critical user transactions.

2. Key Database Entities (Conceptual model)

Since direct DB access is not available, this analysis is based on the conceptual structure of an E-commerce system and verified via front-end actions and network payload inspection.

Entity Name	Description	Key Fields (Conceptual)	Related Transactions
oc_customer	Stores all registered user information and login credentials.	customer_id, firstname, email, password (hashed).	User Registration, User Login, Profile Update.
oc_product	Stores information about the items for sale.	product_id, name, price, quantity (stock level).	Admin Product Creation, Catalog Browsing, Order Placement.
oc_order	Main table recording summary details for every placed order.	order_id, customer_id, total, order_status.	Checkout, Order History View, Admin Order Status Update.
oc_order_product	Links orders to specific products, quantity, and price at the time of purchase.	order_id, product_id, quantity.	Checkout, Order History View.

3. Data Constraints for Testing

These are constraints that must be verified through input validation and data storage integrity checks:

- **Unique Constraints**: The email field in oc_customer must be unique. Testing must include attempts to register with an already used email.
- **Length Constraints:** Test boundary values for fields like firstname, lastname, and password to ensure the system handles min and max lengths correctly.
- **Format Constraints:** Test fields like email for correct format using invalid inputs (e.g., missing @ or .com).
- **Foreign Key Integrity:** When an order is created (in oc_order), the customer_id must successfully reference an existing record in oc_customer.

4. Data Integrity Test Cases

These test cases ensure that data modifications are performed accurately across related tables.

Test Case 1: New User Registration Integrity (Constraint Verification)

- **Action:** Attempt to register a new user using a valid email and then attempt a second registration with the same email.
- **Verification:** The first attempt must succeed, creating a record. The second must fail (Unique Constraint) with appropriate error message on the UI.
- **Expected Result:** Appropriate UI error message for constraint violation is displayed; only one customer record is created.

Test Case 2: Order Status and Stock Integrity

- **Action:** Complete a full checkout successfully, reducing stock of Product X by 1 unit. Then, using Admin, change the Order Status to "Complete".
- **Verification:** Verify the order_status in the conceptual oc_order table is updated. Crucially, verify that the stock of Product X in oc_product is not decremented again upon status change.
- **Expected Result:** Stock level remains consistent after the status update; status change is reflected accurately.

Test Case 3: Stock Consistency After Order Cancellation

- **Action:** Place a successful order for Product Y (Stock = 5) reducing stock to 4. Then, using the Admin panel, change the Order Status to "Canceled" (or a similar status that triggers a stock refund).
- **Verification:** Verify the order_status in the conceptual oc_order table is updated. Crucially, verify that the stock of Product Y in oc_product is correctly incremented back to its original value (5).
- **Expected Result:** The stock level of the canceled product is fully restored to the pre-order amount, and the order status reflects "Canceled."

Test Case 4: Price Integrity Check (Snapshot Testing)

- **Action:**
  1. Note the current price of Product Z ($100) and add it to the cart.
  2. Before checking out, change the price of Product Z via the Admin panel to a new value ($150).
  3. Complete the checkout process using the cart containing Product Z.
- **Verification:** Check the final recorded price in the conceptual oc_order_product table. The recorded price must be $100 (the price at the time the item was added to the order), not the new current price ($150).
- **Expected Result:** The order record maintains the price snapshot at the time of purchase, demonstrating transactional integrity.

Test Case 5: Login and Password Hashing Integrity

- **Action:** Attempt to log in with a valid email and a known valid password, followed by an attempt to log in with the same email but a subtly incorrect password (e.g., wrong capitalization).
- **Verification:** Verify that the first login attempt succeeds, confirming that the system correctly retrieves and compares the submitted password (after hashing) against the stored hash. The second attempt must fail.
- **Expected Result:** The order record maintains the price snapshot at the time of purchase, demonstrating transactional integrity.