Programming Assignment Unit 7

Department of Computer Science, UoPeople

CS 2203-01 - AY2025-T1

Instructor Ada Ajunwa

**Friday, 25th October 2023**

### Database Implementation for an Online Marketplace Platform

**a) Importance of Transactions in Ensuring Data Integrity and Consistency**

Transactions are essential in ensuring the integrity and consistency of data in database systems, particularly in an online marketplace platform where multiple critical operations occur simultaneously. Transactions ensure that a series of operations, such as processing an order or updating inventory levels, are treated as a single unit of work. If any part of the transaction fails, the entire transaction is rolled back, leaving the database in its previous state to avoid inconsistencies. For instance, when a user places an order, various operations occur: deducting the product from inventory, creating a transaction record, and updating the user’s purchase history. If one of these steps fails (e.g., if the inventory update fails due to a system crash), the other steps must also fail (rollback), so that no partial data updates are left in the system. By using transactions, I would ensure that all critical operations related to order processing are either fully completed (committed) or, in case of failure, none are applied (rolled back). This guarantees data accuracy, preventing issues like double-selling products or incorrect order states.

**b) Role of Static, Dynamic, and Embedded SQL**

In the online marketplace platform, SQL (Structured Query Language) plays a significant role in interacting with the database. Static SQL involves SQL queries that are pre-defined and hardcoded in the application. These queries are efficient and secure, as they are compiled and optimized during application development. Static SQL would be beneficial for operations that are repetitive, such as retrieving product categories or customer details, where the SQL queries remain consistent.

Dynamic SQL, on the other hand, is generated and executed at runtime, providing more flexibility for executing complex queries based on user input. In the marketplace context, dynamic SQL would be useful for creating personalized queries, such as searching products by various filters (price, category, etc.). Dynamic SQL allows for more versatility but requires careful handling to prevent security risks like SQL injection attacks.

Embedded SQL refers to SQL queries embedded within the programming code (e.g., Java or Python). This is commonly used to simplify interactions between the application and the database. In the online marketplace, embedded SQL would allow me to integrate SQL statements within the backend code that processes user requests, such as querying the database for available products or updating user reviews.

**c) Utilization of JDBC and ODBC in Database Connections**

JDBC (Java Database Connectivity) and ODBC (Open Database Connectivity) are key technologies that enable an application to connect to and interact with databases.

**Example of JDBC Connection in Java:**

Here’s a simple example of how to establish a connection to a MySQL database using JDBC in Java:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DatabaseConnection {

public static void main(String[] args) {

// Database URL and credentials

String url = "jdbc:mysql://localhost:3306/online\_marketplace";

String user = "root";

String password = "your\_password";

// Establishing the connection

try (Connection connection = DriverManager.getConnection(url, user, password)) {

if (connection != null) {

System.out.println("Connected to the database successfully!");

}

} catch (SQLException e) {

System.out.println("Connection failed: " + e.getMessage());

}

}

}

In this example, the DriverManager class is used to establish a connection to the MySQL database. The connection can then be used to execute SQL queries for operations such as retrieving product listings or processing transactions.

**Example of ODBC Connection in Python:**

For ODBC, here’s an example using Python with the pyodbc library to connect to a database:

import pyodbc

# Define the connection string

connection\_string = 'DRIVER={ODBC Driver 17 for SQL Server};SERVER=localhost;DATABASE=online\_marketplace;UID=user;PWD=password'

# Establishing the connection

try:

connection = pyodbc.connect(connection\_string)

print("Connected to the database successfully!")

except pyodbc.Error as e:

print("Connection failed: ", e)

In this example, the connection string specifies the ODBC driver, server name, database name, and user credentials. This allows the Python application to connect to the SQL Server database, enabling operations such as executing SQL commands to manage products and transactions in the marketplace.

By leveraging JDBC and ODBC, I can ensure that the online marketplace platform maintains a seamless connection with the database, allowing for real-time data processing, efficient query execution, and scalability as the platform grows.

**References**

* Connolly, T., & Begg, C. (2015). Database Systems: A Practical Approach to Design, Implementation, and Management. Pearson.
* Date, C. J. (2019). An Introduction to Database Systems. Pearson.