2300031836 Rushieshwara Reddy

# Skill week 11

1.

CREATE DATABASE LibraryDB;

USE LibraryDB;

CREATE TABLE Books (

book\_id INT PRIMARY KEY,

title VARCHAR(255) NOT NULL,

author VARCHAR(255) NOT NULL,

is\_available BOOLEAN DEFAULT TRUE

);

CREATE TABLE Users (

user\_id INT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL

);

import java.sql.\*;

public class LibraryDB {

private static final String URL = "jdbc:mysql://localhost:3306/LibraryDB";

private static final String USER = "root"; // Change this to your DB username

private static final String PASSWORD = "password"; // Change this to your DB password

public static Connection getConnection() throws SQLException {

return DriverManager.getConnection(URL, USER, PASSWORD);

}

}

public void viewAvailableBooks() {

String query = "SELECT \* FROM Books WHERE is\_available = TRUE";

try (Connection conn = LibraryDB.getConnection();

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(query)) {

System.out.println("Available Books:");

while (rs.next()) {

int bookId = rs.getInt("book\_id");

String title = rs.getString("title");

String author = rs.getString("author");

System.out.println("Book ID: " + bookId + ", Title: " + title + ", Author: " + author);

}

} catch (SQLException e) {

e.printStackTrace();

}

}

public boolean checkoutBook(int userId, int bookId) {

String checkAvailabilityQuery = "SELECT is\_available FROM Books WHERE book\_id = ?";

String checkoutQuery = "UPDATE Books SET is\_available = FALSE WHERE book\_id = ?";

try (Connection conn = LibraryDB.getConnection()) {

try (PreparedStatement checkStmt = conn.prepareStatement(checkAvailabilityQuery)) {

checkStmt.setInt(1, bookId);

try (ResultSet rs = checkStmt.executeQuery()) {

if (rs.next() && rs.getBoolean("is\_available")) {

try (PreparedStatement checkoutStmt = conn.prepareStatement(checkoutQuery)) {

checkoutStmt.setInt(1, bookId);

int rowsUpdated = checkoutStmt.executeUpdate();

if (rowsUpdated > 0) {

System.out.println("Book checked out successfully.");

return true;

}

}

} else {

System.out.println("Book is not available for checkout.");

}

}

}

} catch (SQLException e) {

e.printStackTrace();

}

return false;

}

public boolean returnBook(int userId, int bookId) {

String checkCheckoutQuery = "SELECT is\_available FROM Books WHERE book\_id = ?";

String returnQuery = "UPDATE Books SET is\_available = TRUE WHERE book\_id = ?";

try (Connection conn = LibraryDB.getConnection()) {

try (PreparedStatement checkStmt = conn.prepareStatement(checkCheckoutQuery)) {

checkStmt.setInt(1, bookId);

try (ResultSet rs = checkStmt.executeQuery()) {

if (rs.next() && !rs.getBoolean("is\_available")) {

try (PreparedStatement returnStmt = conn.prepareStatement(returnQuery)) {

returnStmt.setInt(1, bookId);

int rowsUpdated = returnStmt.executeUpdate();

if (rowsUpdated > 0) {

System.out.println("Book returned successfully.");

return true;

}

}

} else {

System.out.println("Book was not checked out or already returned.");

}

}

}

} catch (SQLException e) {

e.printStackTrace();

}

return false;

}

public class LibraryMain {

public static void main(String[] args) {

LibraryDB library = new LibraryDB();

library.viewAvailableBooks();

library.checkoutBook(1, 101);

library.checkoutBook(1, 101);

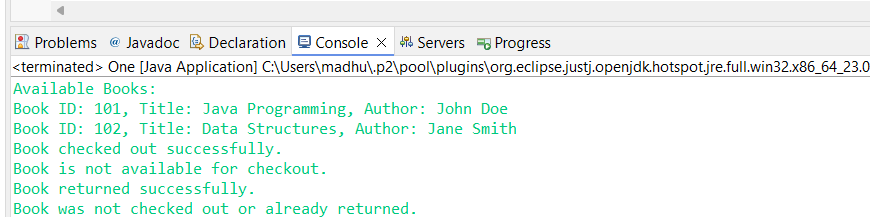
library.returnBook(1, 101);

library.returnBook(1, 101);

}

}

Output:

2.

CREATE DATABASE IF NOT EXISTS EmployeeDB;

USE EmployeeDB;

CREATE TABLE IF NOT EXISTS Employees (

employee\_id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

position TEXT NOT NULL,

salary REAL NOT NULL

);

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DatabaseConnection {

private static final String URL = "jdbc:sqlite:EmployeeDB.db";

private static final String USER = "postgres";

private static final String PASSWORD = "postgres";

public static Connection getConnection() throws SQLException {

return DriverManager.getConnection(URL);

}

}

import java.sql.\*;

public class EmployeeManager {

public void addEmployee(String name, String position, double salary) {

String sql = "INSERT INTO Employees (name, position, salary) VALUES (?, ?, ?)";

try (Connection conn = DatabaseConnection.getConnection();

PreparedStatement stmt = conn.prepareStatement(sql)) {

stmt.setString(1, name);

stmt.setString(2, position);

stmt.setDouble(3, salary);

int rowsInserted = stmt.executeUpdate();

if (rowsInserted > 0) {

System.out.println("Employee added successfully.");

}

} catch (SQLException e) {

System.out.println("Error adding employee: " + e.getMessage());

}

}

public void viewEmployee(int employeeId) {

String sql = "SELECT \* FROM Employees WHERE employee\_id = ?";

try (Connection conn = DatabaseConnection.getConnection();

PreparedStatement stmt = conn.prepareStatement(sql)) {

stmt.setInt(1, employeeId);

try (ResultSet rs = stmt.executeQuery()) {

if (rs.next()) {

int id = rs.getInt("employee\_id");

String name = rs.getString("name");

String position = rs.getString("position");

double salary = rs.getDouble("salary");

System.out.println("Employee ID: " + id);

System.out.println("Name: " + name);

System.out.println("Position: " + position);

System.out.println("Salary: $" + salary);

} else {

System.out.println("Employee not found.");

}

}

} catch (SQLException e) {

System.out.println("Error retrieving employee: " + e.getMessage());

}

}

public void updateEmployee(int employeeId, String name, String position, double salary) {

String sql = "UPDATE Employees SET name = ?, position = ?, salary = ? WHERE employee\_id = ?";

try (Connection conn = DatabaseConnection.getConnection();

PreparedStatement stmt = conn.prepareStatement(sql)) {

stmt.setString(1, name);

stmt.setString(2, position);

stmt.setDouble(3, salary);

stmt.setInt(4, employeeId);

int rowsUpdated = stmt.executeUpdate();

if (rowsUpdated > 0) {

System.out.println("Employee details updated successfully.");

} else {

System.out.println("Employee not found.");

}

} catch (SQLException e) {

System.out.println("Error updating employee: " + e.getMessage());

}

}

public void deleteEmployee(int employeeId) {

String sql = "DELETE FROM Employees WHERE employee\_id = ?";

try (Connection conn = DatabaseConnection.getConnection();

PreparedStatement stmt = conn.prepareStatement(sql)) {

stmt.setInt(1, employeeId);

int rowsDeleted = stmt.executeUpdate();

if (rowsDeleted > 0) {

System.out.println("Employee deleted successfully.");

} else {

System.out.println("Employee not found.");

}

} catch (SQLException e) {

System.out.println("Error deleting employee: " + e.getMessage());

}

}

}

import java.util.Scanner;

public class EmployeeManagementSystem {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

EmployeeManager employeeManager = new EmployeeManager();

while (true) {

System.out.println("\nEmployee Management System");

System.out.println("1. Add Employee");

System.out.println("2. View Employee");

System.out.println("3. Update Employee");

System.out.println("4. Delete Employee");

System.out.println("5. Exit");

System.out.print("Enter your choice: ");

int choice = scanner.nextInt();

scanner.nextLine();

switch (choice) {

case 1:

System.out.print("Enter employee name: ");

String name = scanner.nextLine();

System.out.print("Enter position: ");

String position = scanner.nextLine();

System.out.print("Enter salary: ");

double salary = scanner.nextDouble();

employeeManager.addEmployee(name, position, salary);

break;

case 2:

System.out.print("Enter employee ID to view: ");

int viewId = scanner.nextInt();

employeeManager.viewEmployee(viewId);

break;

case 3:

System.out.print("Enter employee ID to update: ");

int updateId = scanner.nextInt();

scanner.nextLine(); // consume newline

System.out.print("Enter new name: ");

String newName = scanner.nextLine();

System.out.print("Enter new position: ");

String newPosition = scanner.nextLine();

System.out.print("Enter new salary: ");

double newSalary = scanner.nextDouble();

employeeManager.updateEmployee(updateId, newName, newPosition, newSalary);

break;

case 4:

System.out.print("Enter employee ID to delete: ");

int deleteId = scanner.nextInt();

employeeManager.deleteEmployee(deleteId);

break;

case 5:

System.out.println("Exiting...");

scanner.close();

return;

default:

System.out.println("Invalid choice. Please try again.");

}

}

}

}

Output:

A screenshot of a computer

Description automatically generated3.

CREATE DATABASE StudentDB;

USE StudentDB;

CREATE TABLE Students (

student\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(255) NOT NULL,

age INT NOT NULL,

grade VARCHAR(10) NOT NULL

);

import java.sql.\*;

import java.util.\*;

public class StudentDAO {

private static final String URL = "jdbc:mysql://localhost:3306/StudentDB";

private static final String USER = "root";

private static final String PASSWORD = "password";

public static Connection getConnection() throws SQLException {

return DriverManager.getConnection(URL, USER, PASSWORD);

}

public void insertStudent(String name, int age, String grade) {

String sql = "INSERT INTO Students (name, age, grade) VALUES (?, ?, ?)";

try (Connection conn = getConnection(); PreparedStatement stmt = conn.prepareStatement(sql)) {

stmt.setString(1, name);

stmt.setInt(2, age);

stmt.setString(3, grade);

stmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

public List<Map<String, Object>> getAllStudents() {

List<Map<String, Object>> students = new ArrayList<>();

String sql = "SELECT \* FROM Students";

try (Connection conn = getConnection(); Statement stmt = conn.createStatement(); ResultSet rs = stmt.executeQuery(sql)) {

while (rs.next()) {

Map<String, Object> student = new HashMap<>();

student.put("student\_id", rs.getInt("student\_id"));

student.put("name", rs.getString("name"));

student.put("age", rs.getInt("age"));

student.put("grade", rs.getString("grade"));

students.add(student);

}

} catch (SQLException e) {

e.printStackTrace();

}

return students;

}

public Map<String, Object> getStudentById(int studentId) {

Map<String, Object> student = null;

String sql = "SELECT \* FROM Students WHERE student\_id = ?";

try (Connection conn = getConnection(); PreparedStatement stmt = conn.prepareStatement(sql)) {

stmt.setInt(1, studentId);

try (ResultSet rs = stmt.executeQuery()) {

if (rs.next()) {

student = new HashMap<>();

student.put("student\_id", rs.getInt("student\_id"));

student.put("name", rs.getString("name"));

student.put("age", rs.getInt("age"));

student.put("grade", rs.getString("grade"));

}

}

} catch (SQLException e) {

e.printStackTrace();

}

return student;

}

public void updateStudent(int studentId, String name, int age, String grade) {

String sql = "UPDATE Students SET name = ?, age = ?, grade = ? WHERE student\_id = ?";

try (Connection conn = getConnection(); PreparedStatement stmt = conn.prepareStatement(sql)) {

stmt.setString(1, name);

stmt.setInt(2, age);

stmt.setString(3, grade);

stmt.setInt(4, studentId);

stmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

public void deleteStudent(int studentId) {

String sql = "DELETE FROM Students WHERE student\_id = ?";

try (Connection conn = getConnection(); PreparedStatement stmt = conn.prepareStatement(sql)) {

stmt.setInt(1, studentId);

stmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

import java.util.\*;

public class StudentDatabaseApp {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

StudentDAO dao = new StudentDAO();

while (true) {

System.out.println("1. Insert Student");

System.out.println("2. View All Students");

System.out.println("3. View Student By ID");

System.out.println("4. Update Student");

System.out.println("5. Delete Student");

System.out.println("6. Exit");

System.out.print("Choose an option: ");

int choice = scanner.nextInt();

scanner.nextLine();

switch (choice) {

case 1:

System.out.print("Enter name: ");

String name = scanner.nextLine();

System.out.print("Enter age: ");

int age = scanner.nextInt();

scanner.nextLine();

System.out.print("Enter grade: ");

String grade = scanner.nextLine();

dao.insertStudent(name, age, grade);

break;

case 2:

List<Map<String, Object>> students = dao.getAllStudents();

for (Map<String, Object> student : students) {

System.out.println(student);

}

break;

case 3:

System.out.print("Enter student ID: ");

int studentId = scanner.nextInt();

Map<String, Object> student = dao.getStudentById(studentId);

if (student != null) {

System.out.println(student);

} else {

System.out.println("Student not found.");

}

break;

case 4:

System.out.print("Enter student ID to update: ");

int updateId = scanner.nextInt();

scanner.nextLine();

System.out.print("Enter new name: ");

String newName = scanner.nextLine();

System.out.print("Enter new age: ");

int newAge = scanner.nextInt();

scanner.nextLine();

System.out.print("Enter new grade: ");

String newGrade = scanner.nextLine();

dao.updateStudent(updateId, newName, newAge, newGrade);

break;

case 5:

System.out.print("Enter student ID to delete: ");

int deleteId = scanner.nextInt();

dao.deleteStudent(deleteId);

break;

case 6:

scanner.close();

System.out.println("Exiting...");

return;

default:

System.out.println("Invalid choice.");

}

}

}

}

Output:

A screenshot of a computer

Description automatically generated