```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;

public class DatabaseManager {
    private static final String DB_URL = "jdbc:mysql://localhost:3306/mydatabase";
    private static final String USER = "user";
    private static final String PASS = "password";

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int option;
        do {
            System.out.println("\n--- Database Management System ---");
            System.out.println("1. Add User");
            System.out.println("2. Update User");
            System.out.println("3. Delete User");
            System.out.println("4. View All Users");
            System.out.println("5. Add Role");
            System.out.println("6. Assign Role to User");
            System.out.println("7. Remove Role from User");
            System.out.println("8. List All Roles");
            System.out.println("9. Search User by Name");
            System.out.println("10. Search User by Email");
            System.out.println("11. Exit");
            System.out.print("Choose an option: ");
            option = scanner.nextInt();
            switch (option) {
                case 1:
                    addUser(scanner);
                    break;
                case 2:
                    updateUser(scanner);
                    break;
                case 3:
                    deleteUser(scanner);
                    break;
                case 4:
                    viewUsers();
                    break;
                case 5:
                    addRole(scanner);
                    break;
                case 6:
                    assignRole(scanner);
                    break;
                case 7:
                    removeRole(scanner);
                    break;
                case 8:
                    listRoles();
                    break;
                case 9:
                    searchUserByName(scanner);
                    break;
                case 10:
                    searchUserByEmail(scanner);
                    break;
                case 11:
                    System.out.println("Exiting the program...");
                    break;
                default:
                    System.out.println("Invalid option. Please enter a valid number.");
            }
        } while (option != 11);
        scanner.close();
    }

    private static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(DB_URL, USER, PASS);
    }

    private static void addUser(Scanner scanner) {
        System.out.print("Enter user name: ");
        scanner.nextLine(); // clear buffer
        String name = scanner.nextLine();
        System.out.print("Enter email: ");
        String email = scanner.next();
        String sql = "INSERT INTO users (name, email) VALUES (?, ?)";
        try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(sql)) {
            conn.setAutoCommit(false);
            pstmt.setString(1, name);
            pstmt.setString(2, email);
            executeUpdate(pstmt, "User added successfully!");
            conn.commit();
        } catch (SQLException e) {
            System.out.println("Error adding user: " + e.getMessage());
        }
    }

    private static void updateUser(Scanner scanner) {
        System.out.print("Enter user id to update: ");
        int id = scanner.nextInt();
        scanner.nextLine(); // clear buffer
        System.out.print("Enter new email: ");
        String email = scanner.nextLine();
        String sql = "UPDATE users SET email = ? WHERE id = ?";
        try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(sql)) {
            conn.setAutoCommit(false);
            pstmt.setString(1, email);
            pstmt.setInt(2, id);
            executeUpdate(pstmt, "User updated successfully!");
            conn.commit();
        } catch (SQLException e) {
            System.out.println("Error updating user: " + e.getMessage());
        }
    }

    private static void deleteUser(Scanner scanner) {
        System.out.print("Enter user id to delete: ");
        int id = scanner.nextInt();
        String sql = "DELETE FROM users WHERE id = ?";
        try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(sql)) {
            conn.setAutoCommit(false);
            pstmt.setInt(1, id);
            executeUpdate(pstmt, "User deleted successfully!");
            conn.commit();
        } catch (SQLException e) {
            System.out.println("Error deleting user: " + e.getMessage());
        }
    }

    private static void viewUsers() {
        String sql = "SELECT * FROM users";
        try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(sql); ResultSet rs = pstmt.executeQuery()) {
            System.out.println("List of all users:");
            while (rs.next()) {
                int id = rs.getInt("id");
                String name = rs.getString("name");
                String email = rs.getString("email");
                System.out.printf("ID: %d, Name: %s, Email: %s\n", id, name, email);
            }
        } catch (SQLException e) {
            System.out.println("Error retrieving users: " + e.getMessage());
        }
    }

    private static void addRole(Scanner scanner) {
        System.out.print("Enter role name: ");
        scanner.nextLine(); // clear buffer
        String roleName = scanner.nextLine();
        String sql = "INSERT INTO roles (role_name) VALUES (?)";
        try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(sql)) {
            conn.setAutoCommit(false);
            pstmt.setString(1, roleName);
            executeUpdate(pstmt, "Role added successfully!");
            conn.commit();
        } catch (SQLException e) {
            System.out.println("Error adding role: " + e.getMessage());
        }
    }

    private static void assignRole(Scanner scanner) {
        System.out.print("Enter user id for role assignment: ");
        int userId = scanner.nextInt();
        System.out.print("Enter role id to assign: ");
        int roleId = scanner.nextInt();
        String sql = "INSERT INTO user_roles (user_id, role_id) VALUES (?, ?)";
        try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(sql)) {
            conn.setAutoCommit(false);
            pstmt.setInt(1, userId);
            pstmt.setInt(2, roleId);
            executeUpdate(pstmt, "Role assigned to user successfully!");
            conn.commit();
        } catch (SQLException e) {
            System.out.println("Error assigning role: " + e.getMessage());
        }
    }

    private static void removeRole(Scanner scanner) {
        System.out.print("Enter user id for role removal: ");
        int userId = scanner.nextInt();
        System.out.print("Enter role id to remove: ");
        int roleId = scanner.nextInt();
        String sql = "DELETE FROM user_roles WHERE user_id = ? AND role_id = ?";
        try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(sql)) {
            conn.setAutoCommit(false);
            pstmt.setInt(1, userId);
            pstmt.setInt(2, roleId);
            executeUpdate(pstmt, "Role removed from user successfully!");
            conn.commit();
        } catch (SQLException e) {
            System.out.println("Error removing role: " + e.getMessage());
        }
    }

    private static void listRoles() {
        String sql = "SELECT * FROM roles";
        try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(sql); ResultSet rs = pstmt.executeQuery()) {
            System.out.println("List of all roles:");
            while (rs.next()) {
                int roleId = rs.getInt("role_id");
                String roleName = rs.getString("role_name");
                System.out.printf("Role ID: %d, Role Name: %s\n", roleId, roleName);
            }
        } catch (SQLException e) {
            System.out.println("Error listing roles: " + e.getMessage());
        }
    }

    private static void searchUserByName(Scanner scanner) {
        System.out.print("Enter user name to search: ");
        scanner.nextLine(); // clear buffer
        String name = scanner.nextLine();
        String sql = "SELECT * FROM users WHERE name LIKE ?";
        try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, "%" + name + "%");
            try (ResultSet rs = pstmt.executeQuery()) {
                System.out.println("Search results:");
                while (rs.next()) {
                    int id = rs.getInt("id");
                    String email = rs.getString("email");
                    System.out.printf("ID: %d, Name: %s, Email: %s\n", id, name, email);
                }
            }
        } catch (SQLException e) {
            System.out.println("Error searching for user: " + e.getMessage());
        }
    }

    private static void searchUserByEmail(Scanner scanner) {
        System.out.print("Enter email to search for: ");
        scanner.nextLine(); // clear buffer
        String email = scanner.nextLine();
        String sql = "SELECT * FROM users WHERE email LIKE ?";
        try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, "%" + email + "%");
            try (ResultSet rs = pstmt.executeQuery()) {
                System.out.println("Search results:");
                while (rs.next()) {
                    int id = rs.getInt("id");
                    String name = rs.getString("name");
                    System.out.printf("ID: %d, Name: %s, Email: %s\n", id, name, email);
                }
            }
        } catch (SQLException e) {
            System.out.println("Error searching for email: " + e.getMessage());
        }
    }

    private static void executeUpdate(PreparedStatement pstmt, String successMessage) throws SQLException {
        int affectedRows = pstmt.executeUpdate();
        if (affectedRows > 0) {
            System.out.println(successMessage);
            System.out.println("Operation was successful.");
        } else {
            System.out.println("Operation failed, no changes were made.");
        }
    }
}
```