Cập nhật tháng 8 năm 2024

[Bài Đọc] Các thao tác với Transaction

Transaction trong JDBC là một đơn vị công việc logic bao gồm một hoặc nhiều thao tác trên cơ sở dữ liệu. Các thao tác này được xử lý như một nhóm để đảm bảo tính toàn vẹn dữ liệu. Trong JDBC, chúng ta có thể quản lý transaction để thực hiện hoặc huỷ bỏ một loạt các thao tác theo nhu cầu.

1. **Tại sao cần Transaction**

- **Atomicity (Tính nguyên tử):** Một transaction hoặc được thực hiện hoàn toàn hoặc không thực hiện gì cả
- **Consistency (Tính nhất quán):** Cơ sở dữ liệu được đảm bảo duy trì trạng thái hợp lệ sau khi transaction hoàn thành
- **Isolation (Tính cô lập):** Mỗi transaction được thực hiện độc lập, không bị ảnh hưởng bởi các transaction khác
- **Durability (Tính bền vững):** Khi một transaction đã được commit, thay đổi sẽ được lưu trữ vĩnh viễn ngay cả khi hệ thống gặp sự cố

2. **Quản lý Transaction trong JDBC**

- **Chế độ tự động commit**
  - Mặc định, JDBC sử dụng **chế độ tự động commit**, nghĩa là mỗi câu lệnh SQL được thực thi sẽ tự động được **commit** ngay sau khi hoàn tất
  - Tắt tự động commit:

```
Connection connection = DriverManager.getConnection(DB_URL, USER, PASSWORD);
connection.setAutoCommit(false);
```

- **Commit một transaction**
  - **Commit** là hành động xác nhận các thay đổi trong transaction. Sau khi commit, các thay đổi sẽ được lưu vĩnh viễn trong cơ sở dữ liệu
  - Thực hiện commit:

```
connection.commit();
```

- **Rollback một transaction**

- **Rollback** là hành động huỷ bỏ tất cả các thay đổi trong transaction kể từ lần commit gần nhất hoặc từ lúc bắt đầu transaction
- Thực hiện rollback:

```
connection.rollback();
```

3. Các bước quản lý Transaction trong JDBC

- Thiết lập kết nối và tắt chế độ tự động commit:

```
Connection connection = DriverManager.getConnection(DB_URL, USER, PASSWORD);
connection.setAutoCommit(false);
```

- Thực hiện các câu lệnh SQL:

```
Statement statement = connection.createStatement();
statement.executeUpdate("UPDATE accounts SET balance = balance - 500 WHERE id = 1");
statement.executeUpdate("UPDATE accounts SET balance = balance + 500 WHERE id = 2");
```

- Kiểm tra lỗi và commit hoặc rollback:

```
try {
    connection.commit();
} catch (SQLException e) {
    connection.rollback();
    e.printStackTrace();
} finally {
    connection.close();
}
```

4. Lưu ý khi sử dụng Transaction

- **Thứ tự quan trọng:** Phải luôn thực hiện commit hoặc rollback trước khi đóng kết nối
- **Deadlock:** Tránh sử dụng các transaction lâu dài vì có thể gây deadlock
- **Isolation Level:** Cân nhắc sử dụng mức độ cô lập phù hợp để cân bằng giữa hiệu suất và tính toàn vẹn dữ liệu

5. Ví dụ

Chuyển tiền giữa 2 tài khoản:

```java
public class PreparedStatementSelectExample  new *
{
    public static void main(String[] args) throws SQLException  new *
    {
        String url = "jdbc:mysql://localhost:3306/testdb";
        String username = "root";
        String password = "123456";


        try (Connection connection = DriverManager.getConnection(url, username, password)) {
            connection.setAutoCommit(false); // Tắt chế độ auto-commit

            try (PreparedStatement withdraw = connection.prepareStatement(
                    sql: "UPDATE accounts SET balance = balance - ? WHERE id = ?");
                 PreparedStatement deposit = connection.prepareStatement(
                         sql: "UPDATE accounts SET balance = balance + ? WHERE id = ?")) {

                // Trừ tiền từ tài khoản nguồn
                withdraw.setDouble( parameterIndex: 1,  x: 1000);
                withdraw.setInt( parameterIndex: 2,  x: 1);
                withdraw.executeUpdate();

                // Cộng tiền vào tài khoản đích
                deposit.setDouble( parameterIndex: 1,  x: 1000);
                deposit.setInt( parameterIndex: 2,  x: 2);
                deposit.executeUpdate();

                connection.commit(); // Commit transaction
                System.out.println("Transaction committed successfully!");

            } catch (SQLException e) {
                connection.rollback(); // Rollback transaction nếu có lỗi
                System.out.println("Transaction rolled back.");
                e.printStackTrace();
            }
        }
    }
}
```

Link tài nguyên đọc thêm: https://www.javatpoint.com/transaction-management-in-jdbc

Danh sách các bài học