**­­TRƯỜNG CAO ĐẲNG KỸ THUẬT CAO THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

----------- 🕮 ----------



**BÁO CÁO TỰ TÌM HIỂU**

**Đề tài: Sử dụng Statement, PreparedStatements, CallableStatement,Transaction trong JDBC**

**----------------**

**------------------------------**

**Giảng viên hướng dẫn : LÊ HỮU VINH**

**Sinh viên thực hiện: Trần Thị Ngọc Cẩn**

**Lớp : CĐ TH 22DDE**

**Khoá : 2022 – 2025**

**TP. HỒ CHÍ MINH, tháng 5 năm 2024**

**MỤC LỤC**

[I. Định nghĩa về JDBC: 2](#_Toc167630694)

[1. JDBC (Java Database Connectivity): 2](#_Toc167630695)

[2. Các thành phần của JDBC trong Java. 2](#_Toc167630696)

[3. Kiến trúc của JDBC. 3](#_Toc167630701)

[4. Mô hình xây dựng JDBC: 4](#_Toc167630706)

[a. Mô hình 2 lớp: 4](#_Toc167630707)

[b. Mô hình 3 tầng : 4](#_Toc167630708)

[5. Đặc tả từng thành phần trong JDBC khi kết nối dữ liệu. 4](#_Toc167630709)

[6. Các loại JDBC Drivers được sử dụng để kết nối các loại cơ sở dữ liệu 5](#_Toc167630713)

[a. Trình điều khiển loại 1 ( JDBC-ODBC ) 5](#_Toc167630714)

[b. Trình điều khiển loại 2 – trình điều khiển API gốc : 6](#_Toc167630715)

[c. Trình điều khiển loại 3 – trình điều khiển Giao thức mạng: 7](#_Toc167630716)

[d. Trình điều khiển loại 4 – trình điều khiển mỏng: 8](#_Toc167630717)

[II. Kết nối JDBC với hệ quản trị cơ sở dữ liệu MySQL 8](#_Toc167630718)

[1. Cài đặt Driver MySQL : 8](#_Toc167630719)

[2. Giới thiệu MySQL Workbench: 9](#_Toc167630720)

[3. Cài đặt MySQL Workbench: 9](#_Toc167630721)

[4. Thực hiện câu truy vấn tĩnh (Statement): 12](#_Toc167630722)

[5. Thực hiện câu truy vấn động bằng PreparedStatement: 13](#_Toc167630723)

[6. Thực thi truy vấn Store Procedure bằng CallableStatement: 14](#_Toc167630724)

[*7.* Thực hiện truy vấn SQL bằng Transaction. 15](#_Toc167630725)

**DANH MỤC HÌNH ẢNH**

[Hình 1.1 Minh hoạ các thành phần trong JDBC. 2](#_Toc167628779)

[Hình 1.2 Mô hình kiến trúc JDBC 3 tầng 3](#_Toc167628780)

[Hình 1.3 Mô tả trình điều khiển loại 1 của JDBC 5](#_Toc167628781)

[Hình 1.4 Mô tả điều khiển loại 2 của JDBC 6](#_Toc167628782)

[Hình 1.5 Mô tả điều khiển loại 3 của JDBC 7](#_Toc167628784)

[Hình 1.6 Mô tả điều khiển loại 4 của JDBC 8](#_Toc167628785)

[Hình 1.7 Cài đặt Driver MySQL 9](#_Toc167628788)

[Hình 1.8 Thêm file module cho thư viện 9](#_Toc167628790)

[Hình 1.9 Thiết lập tài khoản trên MySQL Workbench 10](#_Toc167628791)

[Hình 2.0 Cơ sở dữ liệu bảng SINHVIEN 11](#_Toc167628792)

[Hình 2.1 Kết nối JDBC với MySQL 11](#_Toc167628793)

[Hình 2.2 Tạo bảng Lop bằng Statement 13](#_Toc167628794)

[Hình 2.3 Kết quả thực hiện câu lệnh Statement 13](#_Toc167628795)

[Hình 2.4 Thêm thông tin bằng PreparedStatement 14](#_Toc167628796)

[Hình 2.5 Kết quả câu truy vấn PreparedStatement 14](#_Toc167628797)

[Hình 2.6 Tạo Store Procedure tìm Lớp theo MaLop 15](#_Toc167628798)

[Hình 2.7 Thực hiện truy vấn CallableStatement 15](#_Toc167628800)

[Hình 2.8 Database số dư An và Bình ban đầu. 16](#_Toc167628803)

[Hình 2.9 Thực hiện truy vấn bằng Transaction 16](#_Toc167628804)

[Hình 3.0 Kết quả sau khi truy vấn bằng Transaction 16](#_Toc167628805)

[Hình 3.1 Thực hiện truy vấn sai trong Transaction 17](#_Toc167628806)

[Hình 3.2 Kết quả truy vấn 17](#_Toc167628807)

[Hình 3.3 Truy vấn sai không dùng Transaction 17](#_Toc167628809)

**GIỚI THIỆU**

Với việc công nghệ ngày càng thay đổi liên tục và được nâng cấp qua từng ngày thì các ngôn ngữ lập trình đóng vai trò quan trọng như một nền tảng cho mọi thứ đổi mới và phát triển. Đó là nền tảng của hàng loạt các công cụ được sử dụng bởi các doanh nghiệp trong mọi lĩnh vực. Vì vậy nên trang bị kiến thức kĩ về một ngôn ngữ lập trình nào đó sẽ là một lợi thế trong thời đại như hiện nay. Và lựa chọn tối ưu nhất đó là Java.

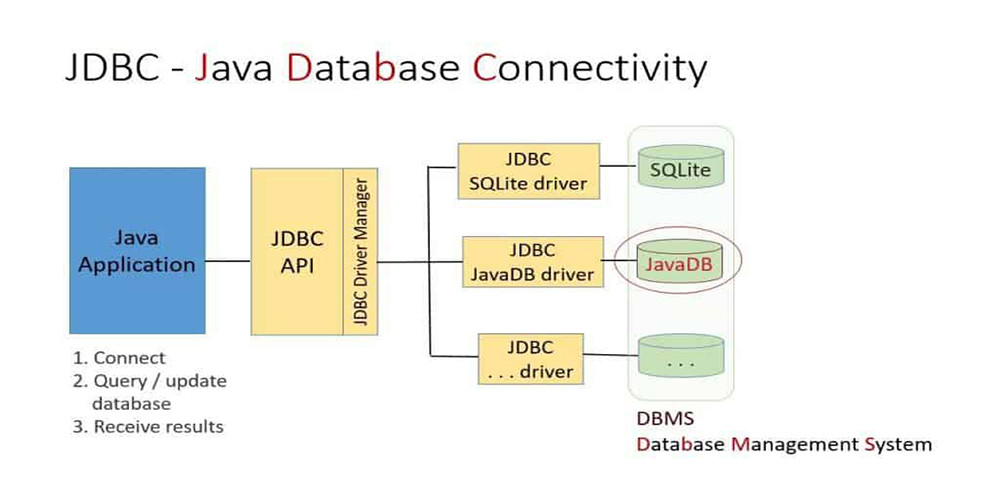
Java là một ngôn ngữ lập trình được phát hành năm 1995 bởi James Gosling ở công ty Sun Microsystems. Tuy đã phát hành rất lâu nhưng Java vẫn được đánh giá là ngôn ngữ trưởng thành, mạnh mẽ và nhiều ứng dụng nhất trong cấc bảng xếp hạng ngôn ngữ qua các năm.

Java là một ngôn ngữ lập trình hiện đại, bậc cao, hướng đối tượng hoàn toàn, bảo mật và mạnh mẽ. Với phương châm của Java là “**Write Once, Run Anywhere**”. Ngôn ngữ này được công nhận về tính linh hoạt trên các nền tảng vì nó hoạt động từ trung tâm dữ liệu máy tính lớn đến điện thoại thông minh. Theo thống kê thì hiện có hơn 3 tỷ thiết bị đang chạy các ứng dụng được xây dựng bằng ngôn ngữ lập trình Java.

Java không những nổi tiếng và nhiều ứng dụng từ tính linh hoạt mà bên cạnh đó Java còn có rất nhiều API, công cụ, thuật toán bảo mật và giao thức trợ giúp. Ngoài ra còn có cả một thư viện ứng dụng đồ sộ giúp cho việc kết nối dữ liệu hoặc tạo giao diện và kể cả là bảo mật cũng được thực hiện dễ dàng và mang lại hiệu quả cao.

Và trong bài báo cáo này em xin phép nói về một trong số tính năng của Java về ứng dụng, xử lí, lưu trữ và truyền dữ liệu được thực hiện bằng phương thức JDBC loại 4 và được minh hoạ bằng việc demo kết nối JDBC bằng MySQL và JDBC loại 4 sẽ hoạt động như thế nào.

1. **Định nghĩa về JDBC:**
2. **JDBC (Java Database Connectivity):**

Là một chuẩn API( Application Program Interface ) cho thực hiện kết nối các chương trình viết bằng Java và các hệ quản trị cơ sở dữ liệu như: MySQL, MS SQL, Oracle, DB2,..

Hình 1.1 Minh hoạ các thành phần trong JDBC.

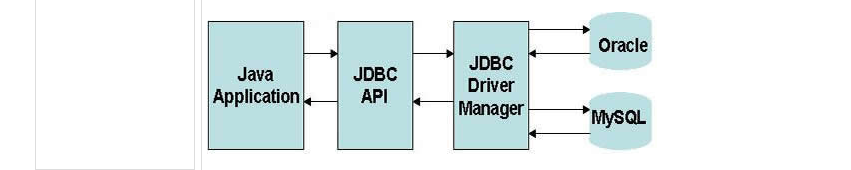
1. **Các thành phần của JDBC trong Java.**
2. ***JDBC API:***

Cung cấp các lớp, phương thức và giao diện khác nhau giúp giao tiếp dễ dàng với cơ sở dữ liệu. Nó cũng cung cấp hai gói chứa nền tảng là: Java SE (***Java Standard Edition***) và Java EE ***( Java Enterprise Edition).***Gói **java.sql** chứa các giao diện và lớp của JDBC API. Nó cung cấp một tiêu chuẩn để kết nối cơ sở dữ liệu với ứng dụng khách.

1. ***JDBC Driver Manager:***

Được dùng để tải trình điều khiển dành riêng cho cơ sở dữ liệu trong một ứng dụng để thiết lập kết nối với cơ sở dữ liệu. Gọi cơ sở dữ liệu cụ thể đến cơ sở dữ liệu để xử lí yêu cầu của người dùng.

1. ***JDBC Test suite:*** Được sử dụng để kiểm tra các hoạt động chẳng hạn như là xoá, thêm ,cập nhật đang được thực hiện bởi **JDBC Drivers.**
2. ***JDBC-ODBC Bridge Drivers:* Kết nối với trình điều khiển cơ sở dữ liệu với cơ sở dữ liệu. Cầu nối này dịch lệnh gọi phương thức JDBC đến lệnh gọi hàm ODBC và sử dụng gói sun.jdbc.odbc bao gồm một thư viện gốc để truy cập các đặc điểm ODBC.**
3. **Kiến trúc của JDBC.**



Hình 1.2 Mô hình kiến trúc JDBC 3 tầng

1. **Applicaion:** Đó là một applet java hoặc là một Servlet giao tiếp với nguồn cơ sở dữ liệu.
2. **JDBC API:** API cho phép chương trình Java thực thi các câu lệnh SQL và truy xuất kết quả. Một số lớp và giao diện quan trọng được định nghĩa trong API JDBC như sau: DriverManager, Driver, Conncetion, Statement, PrepearedStatement, CallableStatement, ResultSet, SQL data.
3. **DriverManager:** Đóng một vai trò quan trọng trong kiến trúc JDBC. Nó sử dụng một số trình điều khiển dành riêng cho cơ sở dữ liệu để kết nối hiệu quả các ứng dụng trong doanh nghiệp với cơ sở dữ liệu.
4. **JDBC Drivers:** Để giao tiếp với nguồn dữ liệu thông qua JDBC, cần trình điều khiển JDBC giao tiếp thông minh với nguồn dữ liệu tương ứng.
5. **Mô hình xây dựng JDBC:**
6. **Mô hình 2 lớp:**

Có thể hiểu một ứng dụng java có thể giao tiếp trực tiếp với nguồn dữ liệu và khi đó trình điều khiển JDBC cho phép giao tiếp giữa ứng dụng và nguồn dữ liệu. Khi người dùng gửi truy vấn đi, câu trả lời cho các đối tượng tuy vấn đó sẽ được gửi lại cho người dùng dưới dạng kết quả.

Nguồn dữ liệu có thể được đặt trên một máy khác trên mạng mà người dùng kết nối. Đây là **cấu hình máy khách/máy chủ**. Trong đó máy thực hiện truy vấn là máy khách và máy gửi câu trả lời là máy chủ.

1. **Mô hình 3 tầng :**

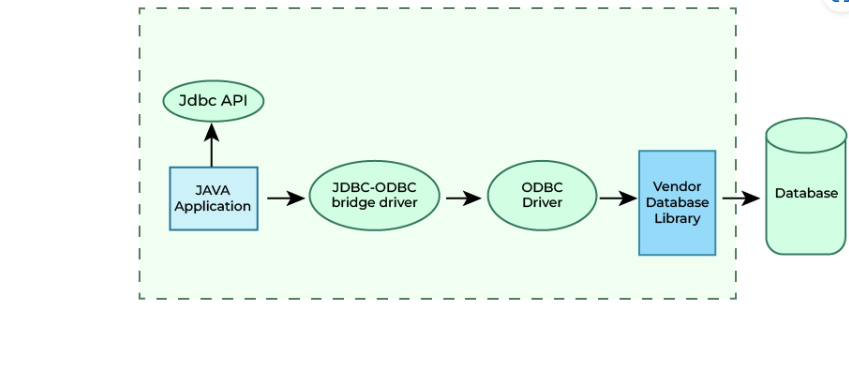
Các truy vấn của người dùng đến được gửi đến các dịch vụ cấp trung, từ đó các lệnh lại được gửi đến nguồn dữ liệu.Kết quả được trả lại tầng giữa và từ đó gửi đến người dùng.

1. **Đặc tả từng thành phần trong JDBC khi kết nối dữ liệu.**
2. **DriverManager:**  Dùng để quản lý danh sách các **Driver** (database drivers).
3. **Driver:** Dùng để liên kết các kết nối tới cơ sở dữ liệu, điều khiển các liên kết.
4. **Connection:** Biểu thị kết nối tới cơ sở dữ liệu. Dùng để tạo Statement, PreparedStatement và CallableStatement. Trong đó:

* **Statement:** Sử dụng để thực hiện các câu truy vấn SQL tĩnh.
* **PreparedStatement:** Sử dụng để thực hiện các truy vấn SQL động hoặc có tham số.
* **CallableStatement:** Sử dụng để thực thi các stored procedures
* **ResultSet:** Biểu diễn một tập kết quả trong cơ sở dữ liệu tạo ra bởi việc sử dụng một câu lệnh SQL là SELECT.
* **SQLException:** Một lớp xử lý lỗi ngoại lệ chứa các lỗi truy cập cơ sở dữ liệu.

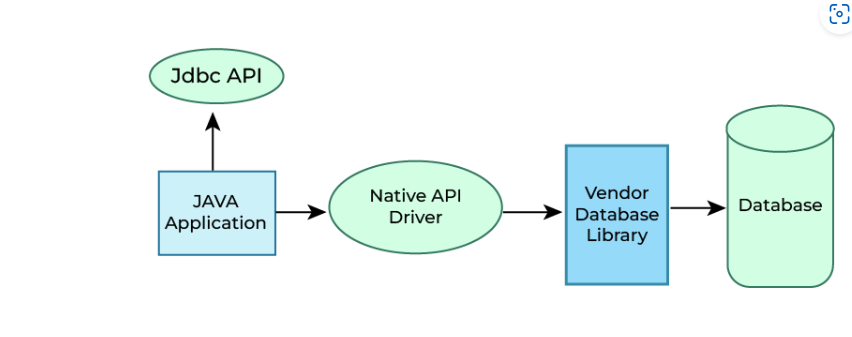
1. **Các loại JDBC Drivers được sử dụng để kết nối các loại cơ sở dữ liệu**
   1. **Trình điều khiển loại 1 ( JDBC-ODBC )**

**ODBC (Open Database Connectivity ):**  Là một API được tạo ra trước JDBC. Mục đích là sử dụng để kết nối và thực hiện truy vấn với cơ sở dữ liệu. Nhưng ODBC được viết bằng ngôn ngữ C từ trước (tức là nền tảng phụ không an toàn trong quá trình kết nối hay chuyển đổi dữ liệu ).

Trình điều khiển cầu nối JDBC-ODBC chuyển đổi các phương thức JDBC thành các lời gọi hàm ODBC. Trình điều khiển này có thể được sử dụng để kết nối với bất kỳ cơ sở dữ liệu nào.

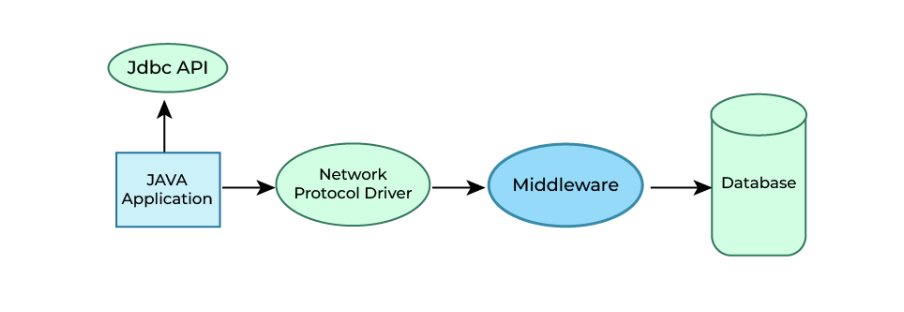
Hình 1.3 Mô tả trình điều khiển loại 1 của JDBC

* **Lợi thế:** điều khiển được tích hợp sẵn với JDK của java và điều khiển độc lập cơ sở dữ liệu.
* **Khó khăn:** Dữ liệu truyền đi không bảo mật vì có thể kết nối với bất kì cơ sở dữ liệu khác. Ngoài ra trình điều khiển ODBC cần được cài đặt trong máy khách cá nhân.
  1. **Trình điều khiển loại 2 – trình điều khiển API gốc :**

Trình điều khiển này cũng sử dụng trình điều khiển ODBC tuy nhiên trình điều khiển này được viết lại bằng Java một phần để cải thiện hiệu suất và khả năng tương thích nên vẫn được gọi là trình điều khiển Java một phần và đó là lí do trình điều khiển này an toàn hơn nhiều so với trình điều khiển loại 1.

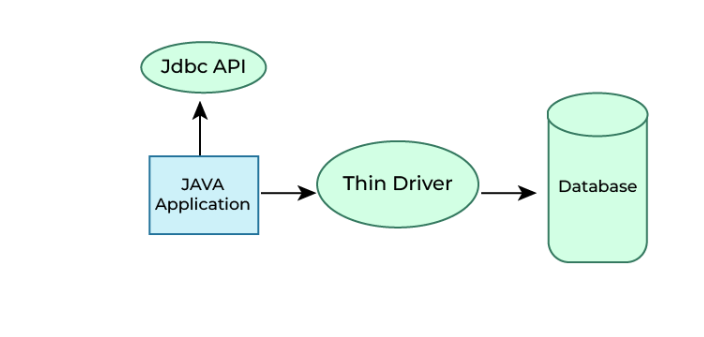
Hình 1.4 Mô tả điều khiển loại 2 của JDBC

* **Khó khăn :** Trình điều khiển và thư viện nhà cung cấpcần được cài đặt riêng mỗi máy khách lẻ. Trình điều khiển này chỉ viết bằng một phần Java nên không được xem như trình điều khiển di dộng. Trình điều khiển này phụ thuộc vào cơ sở dữ liệu kết nối.
  1. **Trình điều khiển loại 3 – trình điều khiển Giao thức mạng:**

Trình điều khiển Giao thức mạng sử dụng các phần mềm trung gian (máy chủ ứng dụng) chuyển đổi các lời gọi JDBC trực tiếp hoặc gián tiếp vào giao thức dành riêng cho nhà cung cấp, trình điều khiển này chỉ sử dụng một máy chủ duy nhất nên không cần cài đặt trên máy khách lẻ.

Hình 1.5 Mô tả điều khiển loại 3 của JDBC

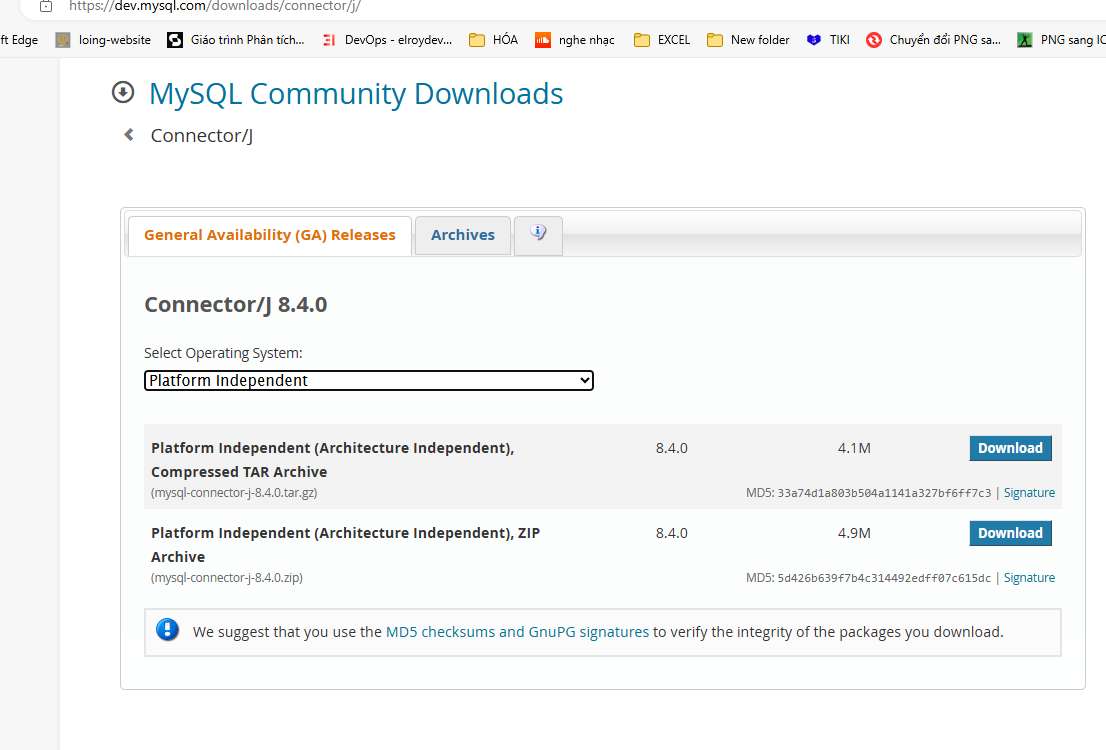
* **Lợi thế:** Trình điều khiển này được viết hoàn toàn bằng Java do đó là trình điều khiển di dộng. Không cần thư viện phía máy khách vì máy chủ có thể thực hiện nhiều tác vụ kiểm tra, cân bằng tải,.. Chuyển đổi cơ sở dữ liệu này sang cơ sở dữ liệu khác.
* **Khó khăn:** Hỗ trợ mạng trên máy khách, bảo trì trình điều khiển Giao thức mạng trở nên tốn kém vì yêu cầu mã hoá dữ liệu từ tầng giữa.
  1. **Trình điều khiển loại 4 – trình điều khiển mỏng:**

Còn được gọi là trình điều khiển giao thức gốc, tương tác trực tiếp với cơ sở dữ liệu. Không yêu cầu bất kì thư viện cơ sở dữ liệu gốc nào đó là lí do gọi trình điều khiển này là trình điều khiển mỏng.

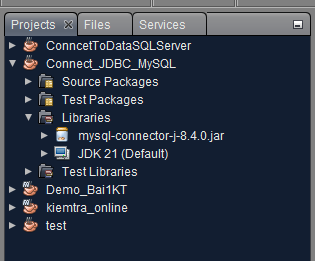
Hình 1.6 Mô tả điều khiển loại 4 của JDBC

* **Lợi thế:** Không yêu cầu bất kì thư viện gốc nào hay máy chủ (Middleware) nào vì vậy không cần cài đặt trên máy khách và máy chủ. Trình điều khiển được viết hoàn toàn bằng Java nên được gọi là trình điều khiển di động.
* **Khó khăn:** Nếu cơ sở dữ liệu thay đổi thì trình điều khiển sẽ mang theo vì phụ thuộc vào cơ sở dữ liệu.

1. **Kết nối JDBC với hệ quản trị cơ sở dữ liệu MySQL**
2. **Cài đặt Driver MySQL :**

Link để tải driver MySQL: [MySQL :: Download Connector/J](https://dev.mysql.com/downloads/connector/j/)

Hình 1.7 Cài đặt Driver MySQL

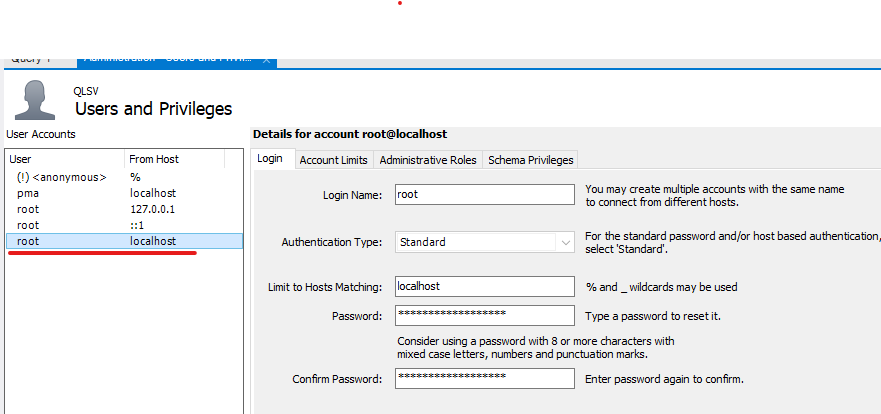
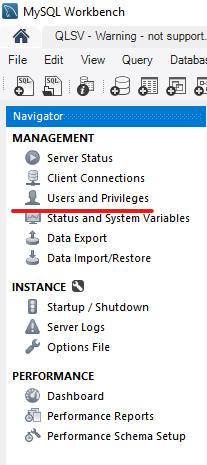
Cài đặt MySQL Driver sau đó vào Netbean tiêns hành tạo một project tên “Connect\_JDBC\_MySQL”. Mở thư mục **Libraries** để xem phiên bản Netbean là thuộc JDK mấy. Có từng phiên bản driver phù hợp cho từng phiên bản JDK khác nhau. Sau đó giải nén Driver vừa tải về chọn vào file \*.**jar** thêm vào bên trong thư mục **Libraries** trong project cá nhân.

Hình 1.8 Thêm file module cho thư viện

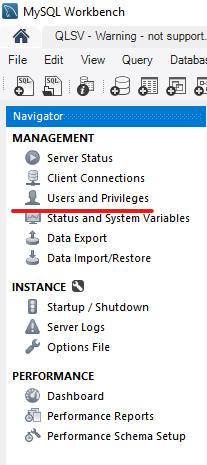
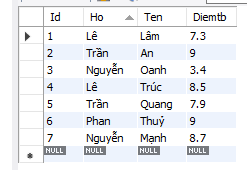
1. **Giới thiệu MySQL Workbench:**

Là công cụ quản lý cơ sở dữ liệu MySQL được phát triển bởi Oracle.Cho phép người dùng thiết kế, truy vấn cũng như quản lý cơ sở dữ liệu MySQL một cách dễ dàng.

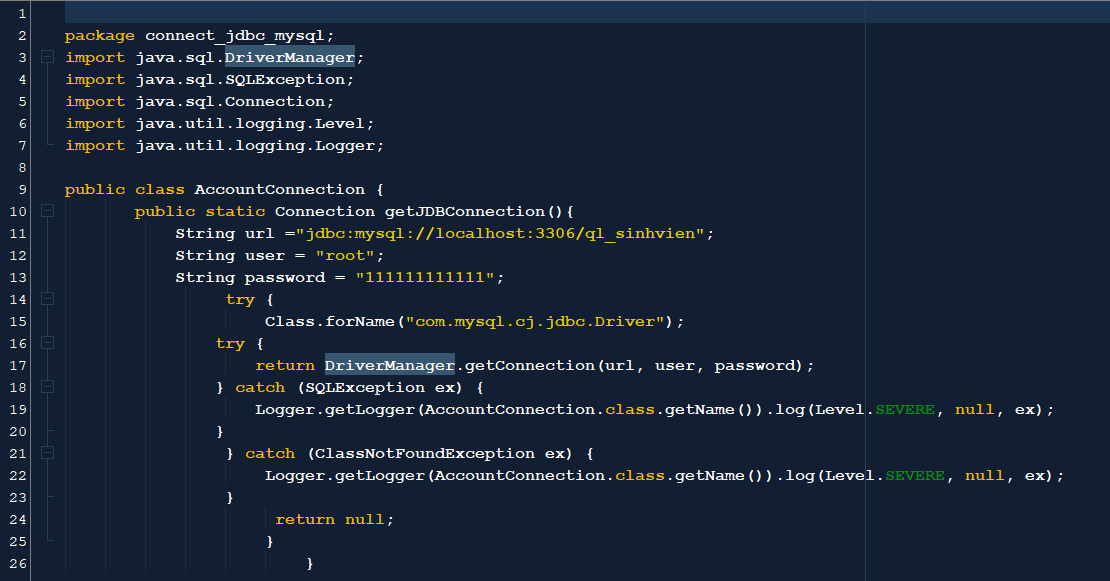
1. **Cài đặt MySQL Workbench:**

Tiến hành cài đặt mật khẩu và tài khoản trong MySQL Workbench. Tài khoản này dùng để thực hiện kết nối với phần mềm Netbeans. Tiến hành tạo một cơ sở dữ liệu trên trang chủ của phần mềm Workbench chọn vào cơ sở dữ liệu vừa tạo.

Hình 1.9 Thiết lập tài khoản trên MySQL Workbench

* Chọn vào Users and Privileges, sau đó giao diện bên cạnh xuất hiện.Chọn vào “root – localhost” sau đó phần login sẽ được hiện ra và thiết lập mật khẩu và tài khoản kết nối sau đó chọn Apply để lưu cài đặt.
* Chọn vào biểu tượng để tạo database mới. Ví dụ có tên database là ql\_sinhvien và bảng SINHVIEN được thiết lập trong bảng chứa dữ liệu sau.

Hình 2.0 Cơ sở dữ liệu bảng SINHVIEN

* Vào Netbeans tạo project tên “Connect\_JDBC\_MySQL”. Tạo kết nối JDBC với MySQL.

Hình 2.1 Kết nối JDBC với MySQL

* Trong đó 3306 là số port khi kết nối cơ sở dữ liệu, “ql\_sinhvien” là tên database kết nối.
* User và password là tên và tài khoản kết nối. Sau đó qua hàm main để gọi hàm getJDBCConnection() và kiểm tra kết quả. Kết quả trả về là thành công.
* **Class.forName**: dùng để chỉ dẫn cho Java biết sẽ kết nối tới loại database nào.
* **url**: Khai báo tạo một kết nối tới database là “ql\_sinhvien” với host mặc định là “localhost” và port là 3306.
* **User**: Trong bài là root lấy mặc định hoặc là 1 user khác khi connection với MySQL Workbench.
* **Password**: Là mật khẩu đăng nhập.

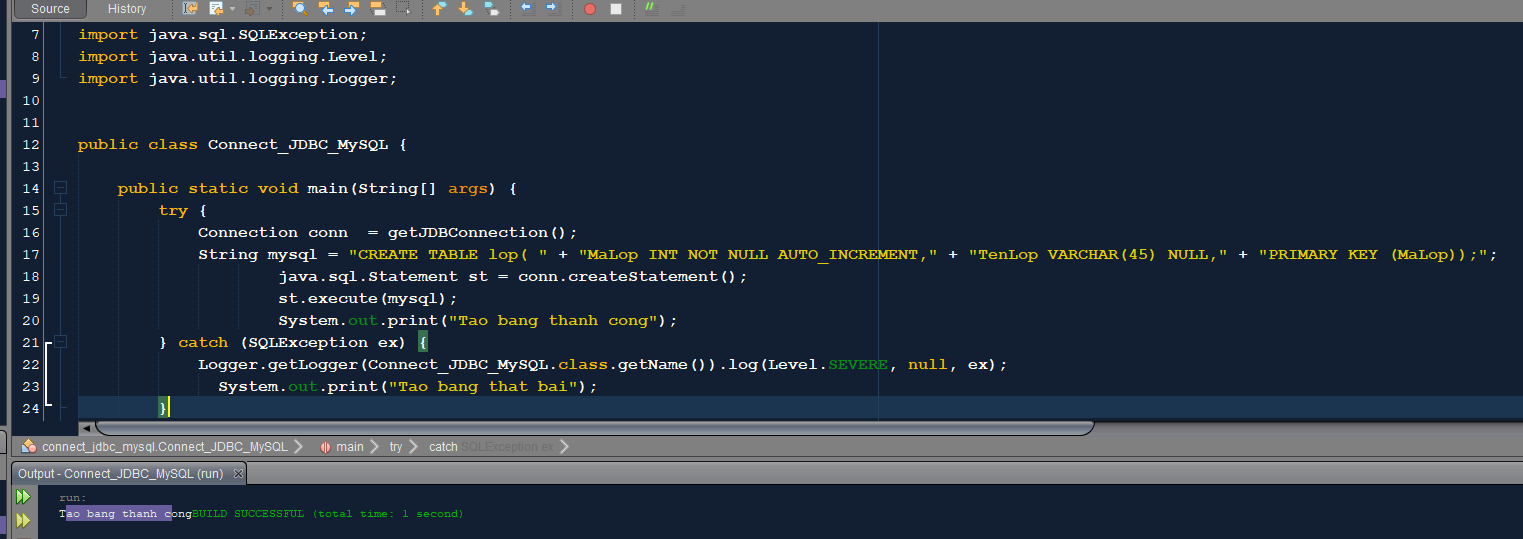
Hoặc có thể kết nối bằng cách gọn hơn là:

* **DriverManager.getConnection(“jdbc:mysql://localhost:3306/ql\_sinhvien”, “root”, “admin1234”):** tạo một connection tới database có url = jdbc:mysql://localhost:3306/ql\_sinhvien với username = “root ” và password  = “1234”
* **Class.forName(“com.mysql.cj.jdbc.Driver”)** biểu thị database MySQL

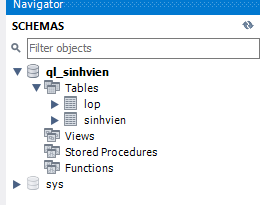
**\*\* Lưu ý:** Sau khi kiểm tra kết nối xong sẽ bắt đầu thực hiện truy vấn bằng các câu lệnh Statement, PreparedStatement, CallableStatement và Transaction. Trước khi sử dụng các câu lệnh thì nên tạo ra các lớp đối tượng và kiểu dữ liệu tương ứng với database.

1. **Thực hiện câu truy vấn tĩnh (Statement):**

* **Statement:**  dùng để thực thi các câu truy vấn tĩnh, không thể truyền tham số vào câu truy vấn SQL trong thơi gian runtime.
* **Statement** có hiệu năng kém hơn so với PrepareStatement và CallableStatement.
* **Statement** chỉ thường được sử dụng trong câu lệnh SQL chỉ chạy 1 lần như: CREATE, ALTER , DROP….

**Vd:** Tạo thêm bảng Lop trong database “ql\_sinhvien” bằng câu lệnh Statement.

Hình 2.2 Tạo bảng Lop bằng Statement

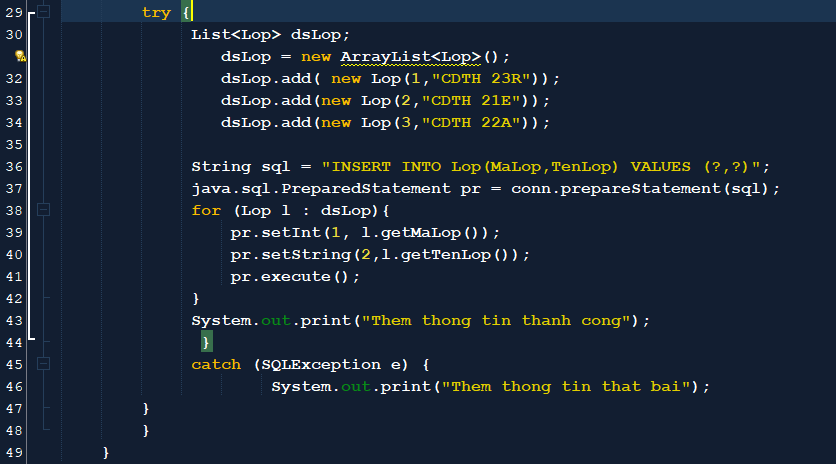
 Và kết quả bên MySQL Workbench:

Hình 2.3 Kết quả thực hiện câu lệnh Statement

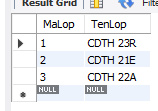
1. **Thực hiện câu truy vấn động bằng PreparedStatement:**

* **PreparedStatement** được sử dụng để thực thi các câu truy vấn SQL động hoặc có tham số.
* **PreparedStatement** kế thừa từ Statement nhưng cho phép truyền các tham số vào câu SQL trong thời gian runtime.
* **PreparedStatement** thích hợp sử dụng cho câu lệnh SQL sử dụng nhiều lần.

**Vd:** Thêm thông tin bảng Lop.

Thực hiện truy vấn:

Hình 2.4 Thêm thông tin bằng PreparedStatement

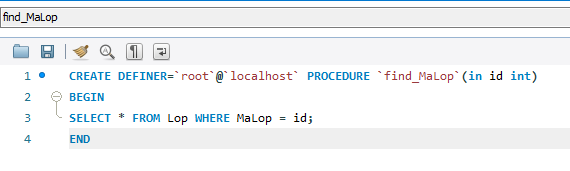
**Kết Quả:**

Hình 2.5 Kết quả câu truy vấn PreparedStatement

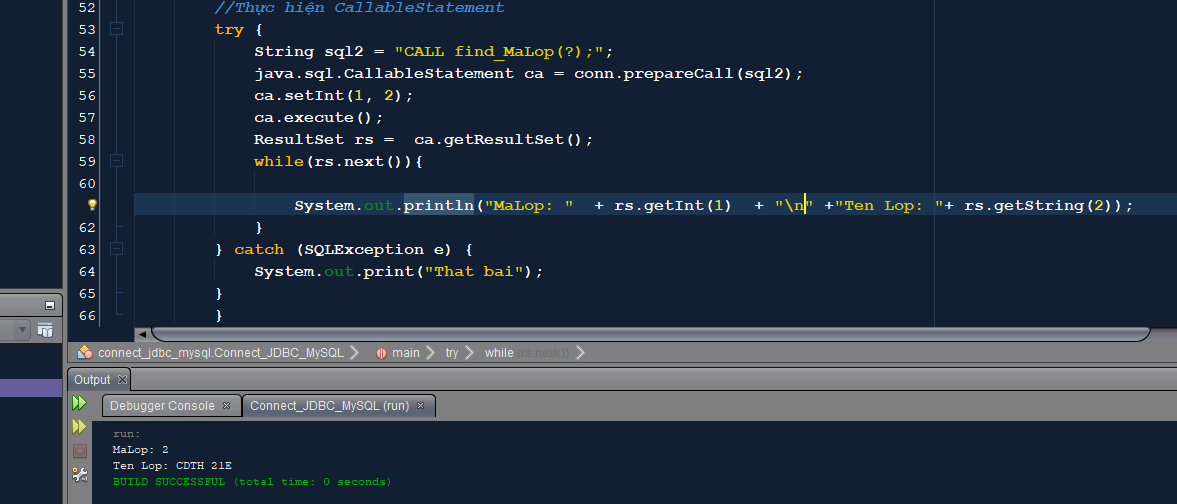
1. **Thực thi truy vấn Store Procedure bằng CallableStatement:**

* **CallableStatement** được sử dụng để thực thi các Store Procedure.
* **CallableStatement** được kế thừa từ PreparedStatement và có hiệu năng cao hơn PreparedStatement.
* **CallableStatement** có thể truyền 3 loại tham số vào Store Procedure là **IN** (truyền giá trị vào), **OUT (**lấy kết quả trả về từ thủ tục**),**  và cả **IN OUT (**thực thi cả IN và OUT**).**

**Vd:** Tìm Lớp học theo MaLop

* Tạo Store Procedure bên MySQL Workbench

Hình 2.6 Tạo Store Procedure tìm Lớp theo MaLop

Thực hiện truy vấn:

Hình 2.7 Thực hiện truy vấn CallableStatement

**\*\*Lưu ý: ResultSet** có chức năng lấy từ câu lệnh “SELECT \* FROM Lop”. Result sẽ lấy hết tất cả các thuộc tính sau khi truy vấn được bằng câu lệnh SELECT.

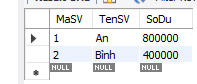
Và rs.getInt() và rs.getString() là lấy kiểu dữ liệu tương ứng để lấy giá trị từ thuộc tính MaLop và TenLop trong bảng Lop. Số 1,2 là các số tương ứng với thứ tự của thuộc tính khi lấy ra. **ResuleSet** tương tự như DataReader bên trong C#.

1. **Thực hiện truy vấn SQL bằng Transaction.**

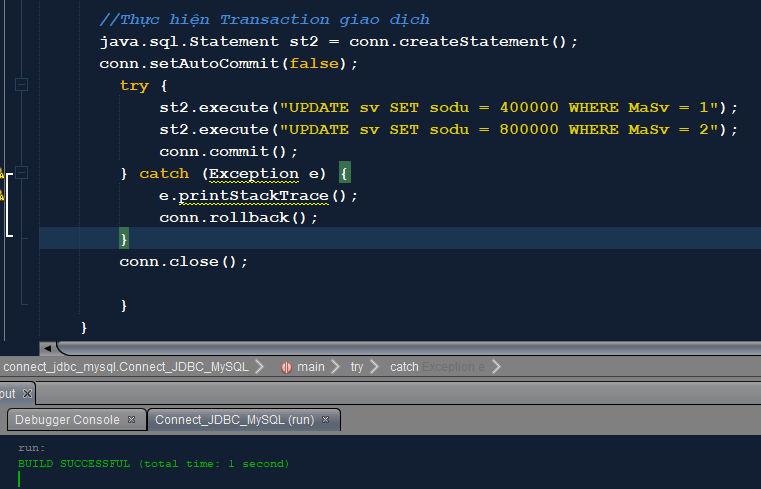
* Là một giao dịch bao gồm một loạt hành động phải được thực hiện cùng nhau và thực hiện thành công hoặc thất bại.
* Nếu một trong hai có hành động thất bại thì tất cả loạt hành động đó sẽ trở về trạng thái ban đầu.
* Được hình dung đơn giản như “Hoặc là có tất cả hoặc không là gì cả”.

Ví dụ về Transaction: Hai sinh viên An và Bình. Khi đó thì chuyển 400.000 VNĐ từ tài khoản của An sang tài khoản của Bình.

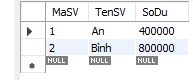
* Tài khoản An: bị trừ 400.000 VNĐ.
* Tài khoản Bình: cộng thêm 400.000 VNĐ.

(*\*\* Giả sử: database đã được kết nối thành công*)

Hình 2.8 Database số dư An và Bình ban đầu.

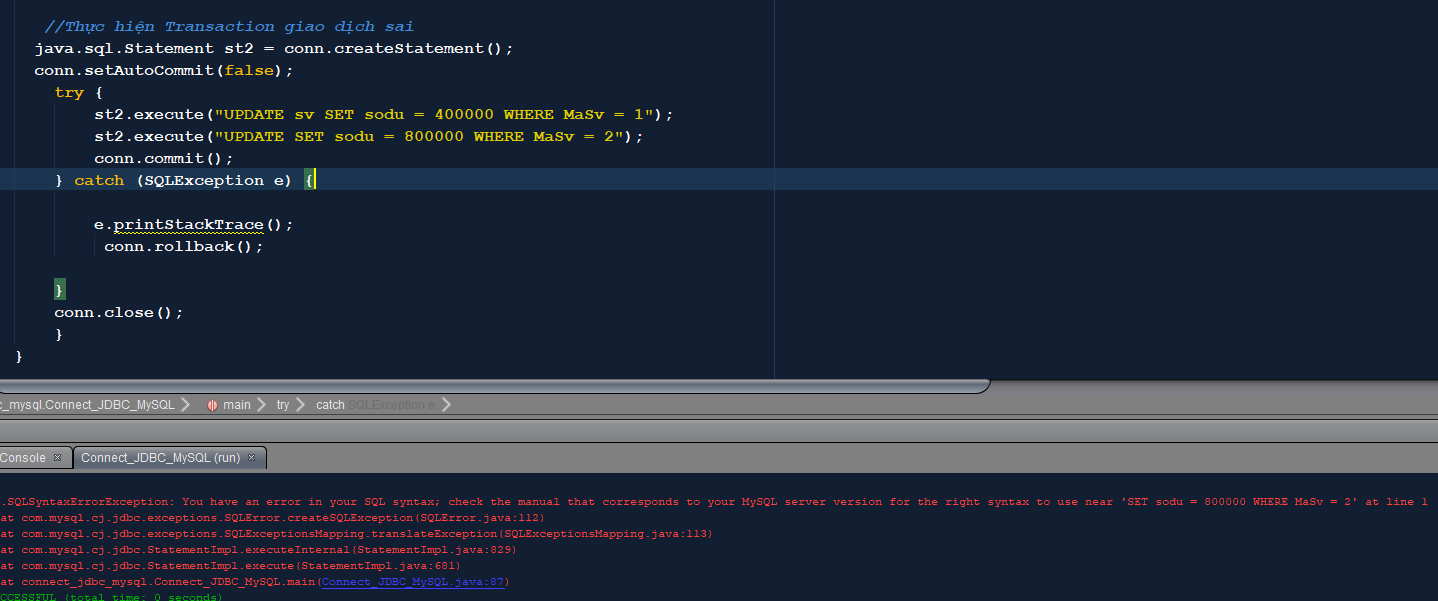
Thực hiện truy vấn:

Hình 2.9 Thực hiện truy vấn bằng Transaction

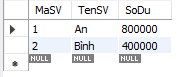
**Kết quả:**

Hình 3.0 Kết quả sau khi truy vấn bằng Transaction

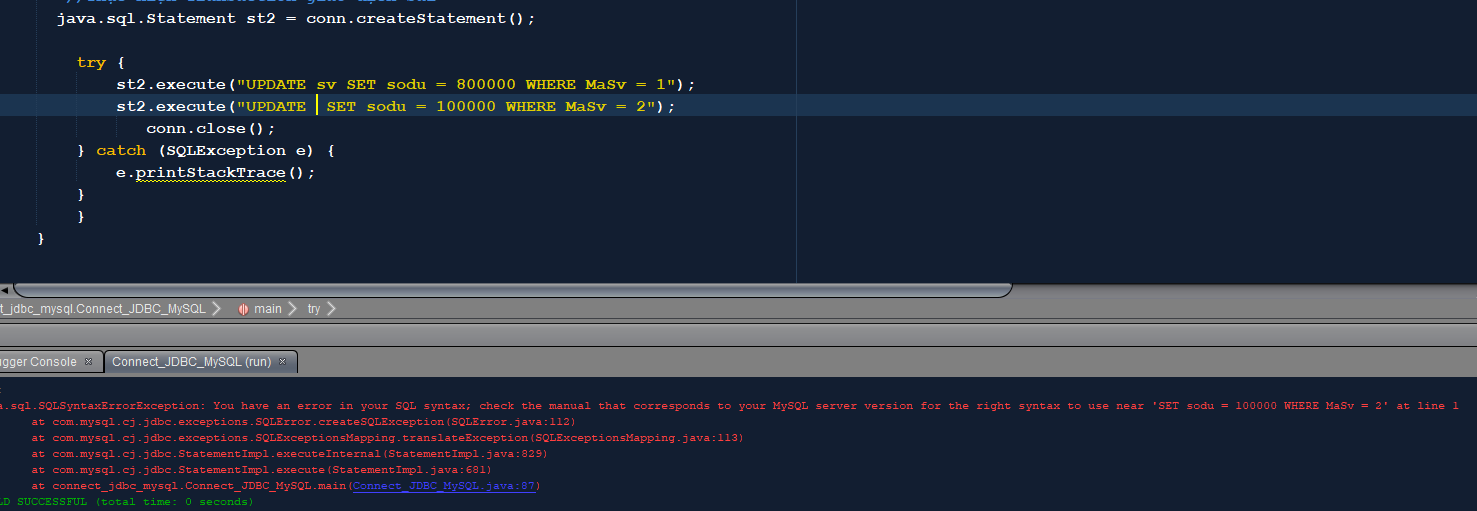
Thực hiện sai trong Transaction: (*\*\* Giả sử:* dữ liệu như ban đầu )



Hình 3.1 Thực hiện truy vấn sai trong Transaction

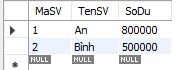
**** **Kết quả:**

Hình 3.2 Kết quả truy vấn

 Thực hiện truy vấn sai khi không có Transaction:

Hình 3.3 Truy vấn sai không dùng Transaction

Kết quả:



**\*\*Lưu ý:** Khi thực hiện Transaction thì loạt hành động này phải thực hiện cùng một lúc.

* **Conn.setAutoCommit(**false**):** Trong một cơ sở dữ liệu luôn đảm bảo tính nhất quán của dữ liệu. Khi một giao dịch cập nhật một hàng trong cơ sở dữ liệu các giao dịch khác không thể thay đổi hàng này cho đến khi giao dịch đầu tiên kết thúc.
* **setAutoCommit**() là phương thức cam kết trong giao dịch. Nếu như nó là **false** thì trình điều khiển JDBC sẽ ngầm bắt đầu một giao dịch mới sau khi cam kết.
* **conn.rollback()** là phương thức khi 1 trong những hành động của loạt giao dịch không thành công. Thì sẽ gọi lại dữ liệu ban đầu và không thực hiện thay đổi cho dù là thành công ở bất kì câu lệnh nào.
* Nếu là (true) thì loạt giao dịch đó sau khi thực hiện xong nó vẫn sẽ giữ kết kết nối như vậy khi có một người khác truy cập và muốn giao dịch tại giá trị đó thì hoàn toàn không được. Vì vậy nên khi sử dụng thì nên setAutocommit(false) để bắt đầu một giao dịch mới và giao dịch này sẽ kết thúc khi gặp lệnh **conn.rollback() hoặc conn.commit()**.