Kết nối cơ sở dữ liệu

SQL Server

Nhóm 4

Thành viên trong nhóm:

Trần Quang Khải

Nguyễn Trường Hùng

Bùi Tiến Đạt

Nguyễn Lê Quang Thuận

Contents

[I. Tầm quan trọng của cơ sở dữ liệu 1](#_Toc110358824)

[II. Gson 1](#_Toc110358825)

[1) Đôi nét về Gson 1](#_Toc110358826)

[2) Một vài ưu điểm của gson 1](#_Toc110358827)

[3) Nhược điểm của Gson 2](#_Toc110358828)

[III. Database 3](#_Toc110358829)

[1) Giới thiệu về Database 3](#_Toc110358830)

[2) Tại sao nên dùng database ? 3](#_Toc110358831)

[IV. SQL-JDBC 4](#_Toc110358832)

[1) Giới thiệu 4](#_Toc110358833)

[2) Ưu điểm 5](#_Toc110358834)

[3) Các thành phần API cơ bản của JDBC 6](#_Toc110358835)

[ DriverManager 6](#_Toc110358836)

[ Driver 7](#_Toc110358837)

[ Connection 8](#_Toc110358838)

[ Statement => Phiên bản tốt hơn: PreparedStatement 9](#_Toc110358839)

[ PreparedStatement 10](#_Toc110358840)

[ ResultSet 12](#_Toc110358841)

1. Tầm quan trọng của cơ sở dữ liệu

Khi source code không có database

* Rất khó để kiểm thử

Không có dữ liệu test hoặc dữ liệu thật

* Giảm mức độ tính toán của hệ thống

Các hệ quản trị CSDL cung cấp các câu query, các function giúp thao tác dữ liệu

* Định hình mô hình ứng dụng

Định hình thiết kế và phát triển ứng dụng sau này

* Không sao lưu được

Chỉ chạy chương trình và mất không lưu và lấy được dữ liệu đã nhập khi tắt chương trình

1. Gson
2. Đôi nét về Gson

* Gson là một thư viện Java mã nguồn mở để tuần tự hóa và giải mã hóa các đối tượng Java thành JSON.

## 2) Một vài ưu điểm của gson

* Dễ sử dụng

Cú pháp JSON rất dễ sử dụng. Chúng ta chỉ phải sử dụng -> như một cú pháp giúp chúng ta phân tích dữ liệu dễ dàng và thực thi dữ liệu nhanh hơn. Vì cú pháp của nó rất nhỏ và trọng lượng nhẹ, đó là lý do mà nó thực thi phản hồi theo cách nhanh hơn.

* Máy chủ phân tích cú pháp

Phân tích cú pháp phía máy chủ là phần quan trọng mà các nhà phát triển muốn nếu quá trình phân tích cú pháp diễn ra nhanh ở phía máy chủ thì người dùng duy nhất có thể nhận được phản hồi nhanh chóng cho phản hồi của họ, vì vậy trong trường hợp này, phân tích cú pháp phía máy chủ JSON là điểm mạnh chỉ ra rằng chúng tôi để sử dụng JSON ở phía máy chủ.

* Hỗ trợ giản đồ

Nó có một loạt các trình duyệt được hỗ trợ tương thích với các hệ điều hành, vì vậy các ứng dụng được tạo bằng mã hóa JSON không đòi hỏi nhiều nỗ lực để làm cho nó tương thích với tất cả các trình duyệt. Trong quá trình phát triển, nhà phát triển nghĩ cho các trình duyệt khác nhau nhưng JSON cung cấp chức năng đó.

* Các đối tượng sắp xếp theo mã

Các đối tượng JSON và các đối tượng mã khớp nhau, điều này có lợi khi tạo nhanh các đối tượng miền trong các ngôn ngữ động.

* Công cụ chia sẻ dữ liệu

Các đối tượng JSON và các đối tượng mã khớp nhau, điều này có lợi khi tạo nhanh các đối tượng miền trong các ngôn ngữ động.

## 3) Nhược điểm của Gson

* Không xử lí lỗi

JSON không xử lý lỗi cho các lệnh gọi JSON. Nếu tính năng chèn tập lệnh động hoạt động, bạn sẽ được gọi và sẽ nhận được phản hồi hoàn hảo. Nếu không được chèn vào, không có gì xảy ra. Nó chỉ thất bại một cách âm thầm. Ví dụ: bạn không thể bắt lỗi 404 từ máy chủ, Bạn cũng không thể hủy hoặc khởi động lại yêu cầu. Tuy nhiên, bạn có thể hết thời gian chờ sau một khoảng thời gian hợp lý.

* Dễ bị tổn thương

JSON có thể khá nguy hiểm nếu được sử dụng với các dịch vụ không đáng tin cậy hoặc trình duyệt không đáng tin cậy, vì dịch vụ JSON trả về một phản hồi JSON được bao bọc trong một lệnh gọi hàm, sẽ được trình duyệt thực thi nếu nó được sử dụng với trình duyệt không đáng tin cậy, nó có thể bị tấn công, điều này khiến ứng dụng Web lưu trữ Dễ bị tấn công bởi nhiều loại tấn công.

* Công cụ hạn chế

Nếu bạn định sử dụng các dịch vụ JSON, điều rất quan trọng là phải nhận thức được các mối đe dọa mà JSON có trong đó và cũng nhận thức được những thứ có thể bảo vệ nó. JSON chỉ có các công cụ được hỗ trợ hạn chế mà chúng tôi có thể sử dụng trong quá trình phát triển JSON.

* Trọng tâm hẹp

Không hỗ trợ định nghĩa ngữ pháp chính thức, do đó, các hợp đồng giao diện rất khó giao tiếp và thực thi. Không hỗ trợ các sản phẩm liên quan đến dịch vụ Web (máy chủ ứng dụng, ESB, v.v.). Không hỗ trợ không gian tên, do đó khả năng mở rộng kém.

1. Database
2. Giới thiệu về Database
   * Đây là thuật ngữ được sử dụng phổ biến trong công nghệ thông tin, dữ liệu, lập trình và phần mềm,… Ngoài ra, nó còn là một bộ sưu tập các dữ liệu được tổ chức một cách bài bản thường được truy cập từ hệ thống máy tính hoặc tồn tại ở dạng tập tin trong hệ quản trị cơ sở dữ liệu.
   * Đặc điểm của cơ sở dữ liệu là có thể truy xuất dữ liệu, thông tin với nhiều cách khác nhau. Các thông tin từ cơ sở dữ liệu đều được đảm bảo về độ an toàn và toàn vẹn dữ liệu nhất, không bị trùng lặp thông tin hoặc có thì tỉ lệ sẽ rất thấp. Hơn hết, Database có thể cho nhiều người dùng sử dụng cùng một lúc.
   * Những cơ sở dữ liệu điển hình nhất có thể kể đến: Danh sách học sinh, bảng chấm công nhân viên, danh sách kiểm kê hàng hoá,…
   * Không chỉ vậy, Database còn được lưu trữ trên những thiết bị có chức năng ghi nhớ như: CD, thẻ nhớ, đĩa cứng,…
3. Tại sao nên dùng database ?

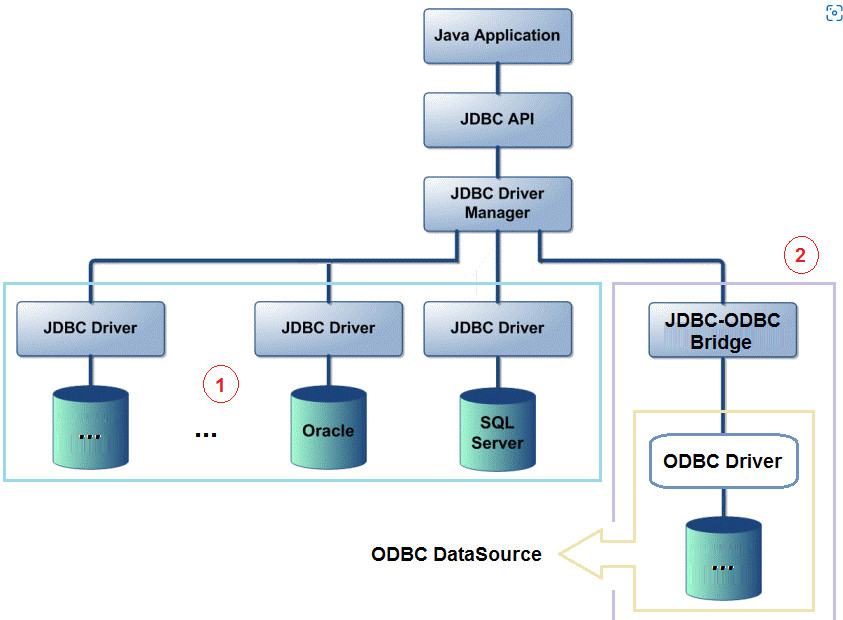
Database là gì? Nó cho phép người dùng có thể truy cập, tìm kiếm các thông tin dữ liệu nhanh chóng và dễ dàng. Trước đây, khi công nghệ thông tin chưa phát triển mạnh mẽ, các thông tin lưu trữ ở trên máy tính dưới dạng tệp tin gây ra nhiều khó khăn cho người sử dụng khi phải tìm kiếm thông tin bằng cách mở từng tệp.

Hiện nay, khi Database ra đời đã giúp khắc phục được các hạn chế mà dữ liệu được lưu dưới dạng tệp tin mắc phải là:

* + Các thông tin lưu trữ sẽ giảm được mức độ trùng lặp xuống mức thấp nhất, tối đa hoá thông tin cần tìm kiếm.
  + Dữ liệu được truy xuất với nhiều cách khác nhau, từ nhiều người dùng khác nhau và từ nhiều ứng dụng khác nhau.
  + Tăng khả năng chia sẻ thông tin với nhiều người dùng mà không gặp các vấn đề về khoảng cách.
  + Bảo mật tối đa thông tin nếu không được chia sẻ từ người dùng chính.

1. SQL-JDBC
2. Giới thiệu

JDBC là một API được thiết kế dành riêng cho ngôn ngữ lập trình java, hỗ trợ java trong việc truy cập cơ sở dữ liệu. Gồm những phương thức hiện truy vấn và cập nhật CSDL gián tiếp qua java.



Ví dụ bạn làm việc với cơ sở dữ liệu Oracle từ Java bạn cần phải có Driver (Đó là class điều khiển việc kết nối với loại cơ sở dữ liệu bạn muốn). Trong JDBC API chúng ta có *java.sql.Driver*, nó chỉ là một interface, và nó có sẵn trong JDK. Như vậy bạn phải download thư viện Driver ứng với loại Database mà bạn mong muốn.

* Chẳng hạn với Oracle thì class thi hành Interface *java.sql.Driver* đó là: oracle.jdbc.driver.OracleDriver

java.sql.DriverManager là một class trong JDBC API. Nó làm nhiệm vụ quản lý các Driver.

Chúng ta có 2 cách để làm việc với một loại cơ sở dữ liệu cụ thể nào đó.

* Cách 1: Bạn hãy cung cấp thư viện Driver điều khiển loại cơ sở dữ liệu đó, đây là cách trực tiếp. Nếu bạn dùng DB oracle (hoặc DB khác) bạn phải download thư viện dành cho loại DB này.
* Cách 2: Khai báo một *"ODBC DataSource"*, và sử dụng cầu nối JDBC-ODBC để kết nối với *"ODBC DataSource"* kia. Cầu nối JDBC-ODBC là thứ có sẵn trong JDBC API.



Câu hỏi của chúng ta là "ODBC DataSource" là cái gì?  
  
ODBC - Open Database Connectivity: Nó chính là một bộ thư viện mở, có khả năng kết nối với hầu hết các loại cơ sở dữ liệu khác nhau, và nó miễn phí. Được cung cấp bởi Microsoft.  
  
ODBC DataSource: Trên hệ điều hành Window bạn có thể khai báo một kết nối ODBC tới một loại DB nào đó. Và như vậy chúng ta có một nguồn dữ liệu (Data Source).  
  
Trong JDBC API, đã xây dựng sẵn một cầu nối JDBC-ODBC để JDBC có thể nói chuyện được với ODBC Data Source.

Về tốc độ, cách 1 sẽ nhanh hơn cách 2, vì cách 2 phải sử dụng tới cầu nối.

Trong trường hợp nếu bạn không muốn sử dụng **JDBC-ODBC**, bạn có thể sử dụng cách trực tiếp kết nối vào Database, trong trường hợp đó cần phải download Driver ứng với mỗi loại DB này.

1. Ưu điểm

* Ngôn ngữ tương tác

SQL là một ngôn ngữ miền được sử dụng để giao tiếp với cơ sở dữ liệu. Nó cũng được sử dụng để nhận câu trả lời cho các câu hỏi phức tạp trong vài giây.

* Không cần mã hóa

Trong SQL tiêu chuẩn, rất dễ dàng để quản lý hệ thống cơ sở dữ liệu. Nó không yêu cầu một lượng lớn mã để quản lý hệ thống cơ sở dữ liệu.

* Tốc độ cao

Sử dụng các truy vấn SQL, người dùng có thể truy xuất một số lượng lớn các bản ghi từ cơ sở dữ liệu một cách nhanh chóng và hiệu quả.

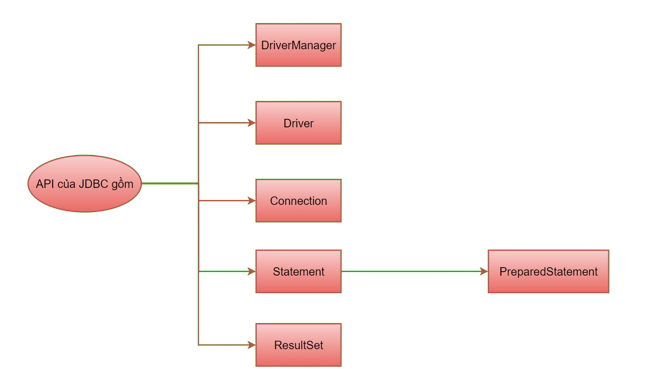
* Tính di động

SQL có thể được sử dụng trong máy tính xách tay, PC, máy chủ và thậm chí một số điện thoại di động.

* Nhiều chế độ xem dữ liệu

Sử dụng ngôn ngữ SQL, người dùng có thể tạo các dạng xem khác nhau của cấu trúc cơ sở dữ liệu.

1. Các thành phần API cơ bản của JDBC



* DriverManager

Lớp DriverManager hoạt động như một giao diện giữa người dùng và các driver. Nó theo dõi các driver có sẵn và xử lý việc thiết lập kết nối giữa một Database và Driver thích hợp.

Một số phương thức phổ biến

1. public static void registerDriver(Driver driver) throws SQLException

Phương thức này được sử dụng để đăng ký driver đã cho với DriverManager.

2. public static void deregisterDriver(Driver driver)

Phương thức này được sử dụng để xóa driver đã cho khỏi danh sách đăng ký với DriverManager.

3. public static Connection getConnection(String url)

Phương thức này được sử dụng để thiết lập kết nối với URL đã cho.

4. public static Connection getConnection(String url,String userName,String password)

Phương thức này được sử dụng để thiết lập kết nối với URL, username, và password đã cho.

5. public static void setLoginTimeout(int second)

Phương thức này thiết lập thời gian (bằng giây) tối đa mà một driver sẽ đợi trong khi cố gắng kết nối với một Database.

6. public static int getLoginTimeout()

Phương thức này lấy thời gian (bằng giây) tối đa mà một driver sẽ đợi trong khi cố gắng đăng nhập vào một Database.

* Driver

Là một Interface, nó dùng để liên kết các liên lạc với cơ sở dữ liệu, điều khiển các liên lạc với database. Một khi Driver được tải lên, lập trình viên không cần phải gọi nó một cách cụ thể.

* Connection

Là một Interface với tất cả các method cho việc liên lạc với database. Nó mô tả nội dung liên lạc. tất cả các thông tin liên lạc với cơ sở dữ liệu là thông qua chỉ có đối tượng Connection.

1. public Statement createStatement()

Phương thức này tạo đối tượng Statement để thực thi các truy vấn SQL.

2. public Statement createStatement(int resultSetType,int resultSetConcurrency) throws SQLException

Phương thức này tạo một đối tượng Statement mà sẽ tạo các đối tượng ResultSet với type và concurrency đã cung cấp.

Tham số resultSetConcurrency là một trong các kiểu ResultSet.CONCUR\_READ\_ONLY hoặc ResultSet.CONCUR\_UPDATABLE.

3. public void setAutoCommit(boolean autoCommit) throws SQLException

Phương thức này thiết lập Connection trong chế độ auto-commit. Nếu một Connection trong chế độ tự động ký thác thì tất cả các lệnh SQL của nó sẽ được thực thi và ký thác sau mỗi giao tác. Theo mặc định thì các Connection mới là trong chế độ auto-commit. Nếu tham số autoCommit được thiết lập là true tức là kích hoạt chế độ auto-commit, nếu là false là vô hiệu hóa chế độ này.

4. public void commit() throws SQLException

Phương thức này lưu các thay đổi đã được thực hiện trước đó. Phương thức này nên chỉ được sử dụng khi chế độ auto-commit đã bị vô hiệu hóa.

5. public void rollback()

Phương thức này xóa tất cả các thay đổi đã được thực hiện trước đó và quay về trạng thái trước khi thực hiện thay đổi. Phương thức này được gọi trên một kết nối đã được đóng hoặc đối tượng Connection này là trong chế độ auto-commit.

6. public void close()

Phương thức này đóng kết nối và giải phóng resource ngay lập tức thay vì chờ đợi chúng tự động được giải phóng. Phương thức này sẽ không hoạt động nếu bạn gọi nó trên một đối tượng đã bị đóng.

7. setSavepoint (String ten) throws SQLException

Phương thức này tạo một savepoint với tên ten đã cho trong giao tác hiện tại và trả về đối tượng Savepoint mới biểu diễn nó. Ở đây ten là tên của savepoint.

* Statement => Phiên bản tốt hơn: PreparedStatement

Là một Interface, gói gọn một câu lệnh SQL gửi tới cơ sở dữ liệu được phân tích, tổng hợp, lập kế hoạch và thực hiện.

1. public ResultSet executeQuery(String sql)

Thực thi một lệnh SQL đã cho và trả về một đối tượng Resultset đơn. Tham số sql là một lệnh SQL.

Phương thức này không thể được gọi trên một đối tượng PreparedStatement hoặc CallableStatement.

2. public int executeUpdate(String sql)

Thực thi lệnh SQL đã cho, có thể là INSERT, UPDATE, DELETE hoặc một lệnh SQL mà không trả về bất kỳ cái gì như lệnh SQL DDL.

Phương thức này không thể được gọi trên một đối tượng PreparedStatement hoặc CallableStatement.

3. public boolean execute(String sql)

Thực thi lệnh SQL đã cho mà có thể trả về nhiều kết quả. Thường thì bạn có thể bỏ qua phương thức này, trừ khi bạn đang thực thi một store procedure mà bạn biết rằng nó sẽ trả về nhiều kết quả hoặc bạn đang thực thi một chuỗi SQL mà bạn chưa biết.

Phương thức này không thể được gọi trên một đối tượng PreparedStatement hoặc CallableStatement.

4. public int[] executeBatch()

Được sử dụng để thực thi một nhóm các lệnh và nếu thành công thì trả về một mảng. Các phần tử trong mảng trả về được sắp xếp theo thứ tự tương ứng với lệnh trong batch. Các phần tử trong mảng được trả về này có thể là:

Một số >= 0, chỉ rằng lệnh được thực thi thành công và đó là số hàng bị tác động trong cơ sở dữ liệu.

Một giá trị SUCCESS\_NO\_INFO chỉ rằng lệnh được thực thi thành công nhưng số hàng bị tác động là chưa biết.

Một giá trị EXECUTE\_FAILED chỉ rằng lệnh đã thất bại.

5. void close() throws SQLException

Đóng đối tượng Statement và giải phóng resource ngay lập tức. Khi đã đóng đối tượng Statement rồi, thì các lời gọi phương thức nào tới đối tượng đó sẽ không hoạt động. Khi một đối tượng Statement đã bị đóng thì đối tượng ResultSet của nó cũng bị đóng.

* PreparedStatement

ResultSet đại diện cho tập hợp các bản ghi lấy do thực hiện truy vấn.

1. public void setInt(int paramIndex, int giaTri)

Thiết lập tham số đã cho tới giá trị nguyên trong Java đã cung cấp. Driver sẽ chuyển đổi giá trị này thành một giá trị nguyên trong SQL khi nó gửi giá trị tới Database.

Trong đó, paramIndex là tham số đầu tiên, tham số thứ hai, …; giaTri là giá trị của tham số.

2. public void setString(int paramIndex, String giaTri)

Thiết lập tham số đã cho thành giá trị String trong Java đã cung cấp. Driver chuyển đổi giá trị này thành một kiểu VARCHAR hoặc LONGVARCHAR (tùy thuộc vào kích cỡ tham số) khi nó gửi giá trị tới Database. (Miêu tả về tham số như trên.)

3. public void setFloat(int paramIndex, float giaTri)

Thiết lập tham số đã cho thành giá trị float trong Java đã cung cấp. Driver chuyển đổi giá trị này thành một giá trị REAL trong SQL khi nó gửi giá trị tới Database. (Miêu tả về tham số như trên.)

4. public void setDouble(int paramIndex, double giaTri)

Thiết lập tham số đã cho thành giá trị double trong Java đã cung cấp. Driver chuyển đổi giá trị này thành một giá trị DOUBLE trong SQL khi nó gửi giá trị tới Database. (Miêu tả về tham số như trên.)

5. public int executeUpdate()

Thực thi truy vấn SQL trong đối tượng PreparedStatement này, mà phải là một lệnh DML (lệnh thao tác dữ liệu) trong SQL như INSERT, UPDATE hoặc DELETE, hoặc một lệnh SQL mà không trả về bất cứ cái gì, chẳng hạn như một lệnh DDL (như CREATE, ALTER, …)

6. public ResultSet executeQuery() throws SQLException

Thực thi truy vấn SQL trong đối tượng PreparedStatement này và trả về đối tượng ResultSet được tạo bởi truy vấn. Phương thức này ném ngoại lệ SQLException và SQLTimeoutException.

* ResultSet

Loại 1: Các phương thức được sử đụng để di chuyển con trỏ bên trong Result Set..

1. public void beforeFirst() throws SQLException

Di chuyển con trỏ về trước hàng đầu tiên.

2. public void afterLast() throws SQLException

Di chuyển con trỏ về sau hàng cuối cùng.

3. public boolean first() throws SQLException

Di chuyển con trỏ về hàng đầu tiên.

4. public void last() throws SQLException

Di chuyển con trỏ về hàng cuối cùng.

5. public boolean absolute(int row) throws SQLException

Di chuyển con trỏ tới hàng đã cho.

6. public boolean relative(int row) throws SQLException

Di chuyển con trỏ tới hàng đã cho (trước hoặc sau) tính từ vị trí hiện tại.

7. public boolean previous() throws SQLException

Di chuyển con trỏ tới hàng trước đó. Phương thức trả về false nếu không có hàng ở trước.

8. public boolean next() throws SQLException

Di chuyển con trỏ tới hàng tiếp theo. Phương thức trả về false nếu không có hàng tiếp theo.

9. public int getRow() throws SQLException

Trả về số hàng mà con trỏ đang trỏ tới.

10. public void moveToInsertRow() throws SQLException

Di chuyển con trỏ tới một hàng đặc biệt trong ResultSet mà có thể được sử dụng để chèn một hàng mới vào trong cơ sở dữ liệu. Vị trí con trỏ hiện tại được ghi nhớ.

**Loại 2: Các phương thức để quan sát dữ liệu trong các cột của hàng hiện tại đang được trỏ bởi con trỏ.**

1. public int getInt(int chi\_muc\_cot)

Trả về dữ liệu của chỉ mục cột đã cho của hàng hiện tại dưới dạng số nguyên.

2. public int getInt(String ten\_cot)

Trả về dữ liệu của tên cột đã cho của hàng hiện tại dưới dạng số nguyên.

3. public String getString(int chi\_muc\_cot)

Trả về dữ liệu của chỉ mục cột đã cho của hàng hiện tại dưới dạng String.

4. public String getString(String ten\_cot)

Trả về dữ liệu của tên cột đã cho của hàng hiện tại dưới dạng String.

5. public Array getArray(int chi\_muc\_cot)

Trả về dữ liệu của chỉ mục cột đã cho của hàng hiện tại dưới dạng một Mảng.

6. public Array getArray(int ten\_cot)

Trả về dữ liệu của tên cột đã cho của hàng hiện tại dưới dạng một Mảng.

**Loại 3: Các phương thức được sử dụng để cập nhật dữ liệu trong các cột của hàng hiện tại.**

1. public void updateInt(int chi\_muc\_cot, int x) throws SQLException

Cập nhật giá trị của cột trong hàng hiện tại hoặc trong hàng đang được chèn tới giá trị nguyên x.

2. public void updateInt(int ten\_cot, int x) throws SQLException

Cập nhật giá trị của tên cột đã cho trong hàng hiện tại hoặc hàng đang được chèn tới giá trị nguyên x.

3. public void updateString(int chi\_muc\_cot, String x) throws SQLException

Cập nhật giá trị của cột trong hàng hiện tại hoặc trong hàng đang được chèn tới giá trị x ở dạng chuỗi.

4. public void updateString(int ten\_cot, String x)

Cập nhật giá trị của tên cột đã cho trong hàng hiện tại hoặc hàng đang được chèn tới giá trị x ở dạng chuỗi.

---Hết---