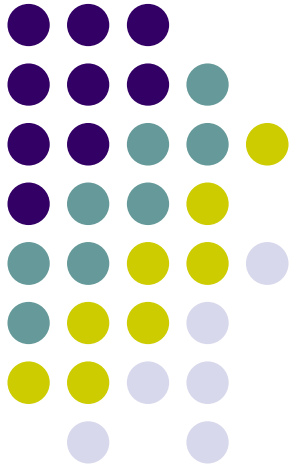
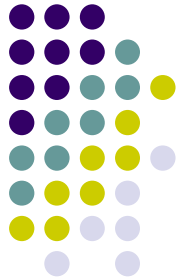
JDBC Căn Bản

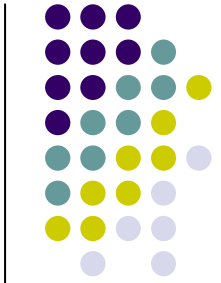


Nội Dung



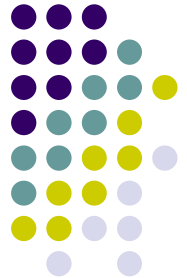
- Giới thiệu JDBC
- Các kiến trúc ứng dụng trên cơ sở dữ liệu
- Kết nối vào cơ sở dữ liệu
- Thực thi các câu lệnh SQL qua JDBC
- Sử dụng mô hình giao dịch trên cơ sở dữ liệu

Giới Thiệu JDBC

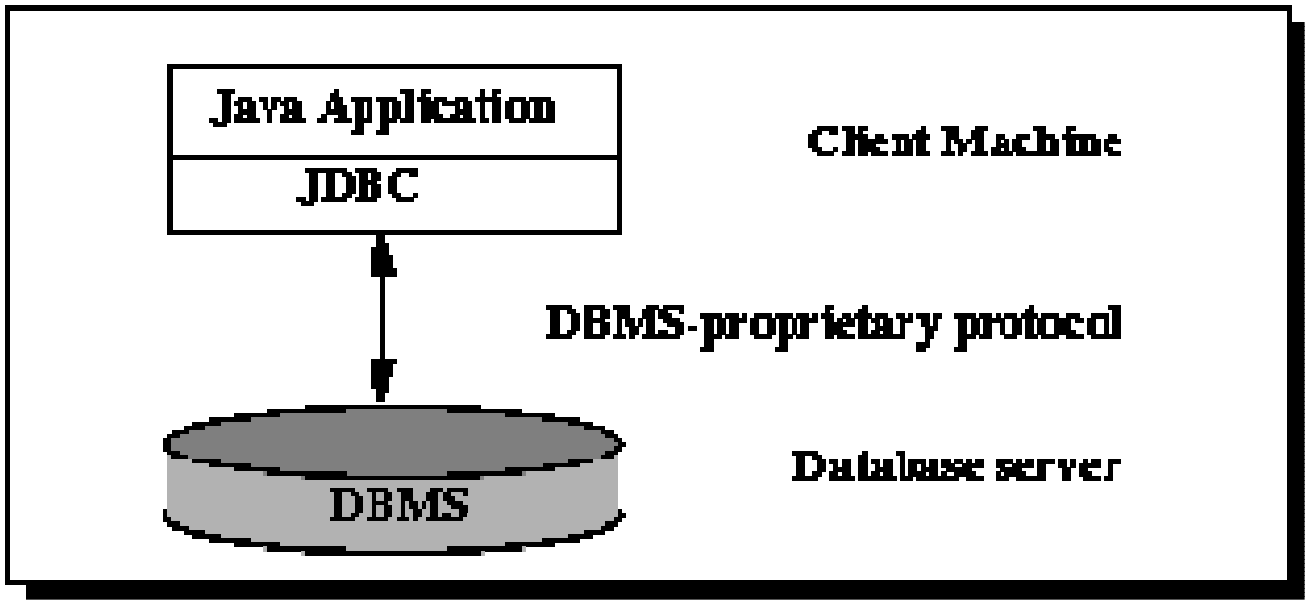


- JDBC: Java Database Connectivity
- Một tập các API trên Java để kết nối đến các hệ quản trị cơ sở dữ liệu
- Các interface và class chủ yếu của JDBC nằm trong gói java.sql
- Một vài interface và class có các chức năng nâng cao nằm trong gói javax.sql

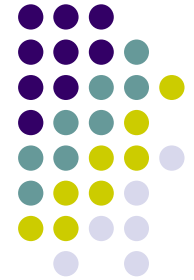
Kiến Trúc Ứng Dụng Sử Dụng JDBC (1)



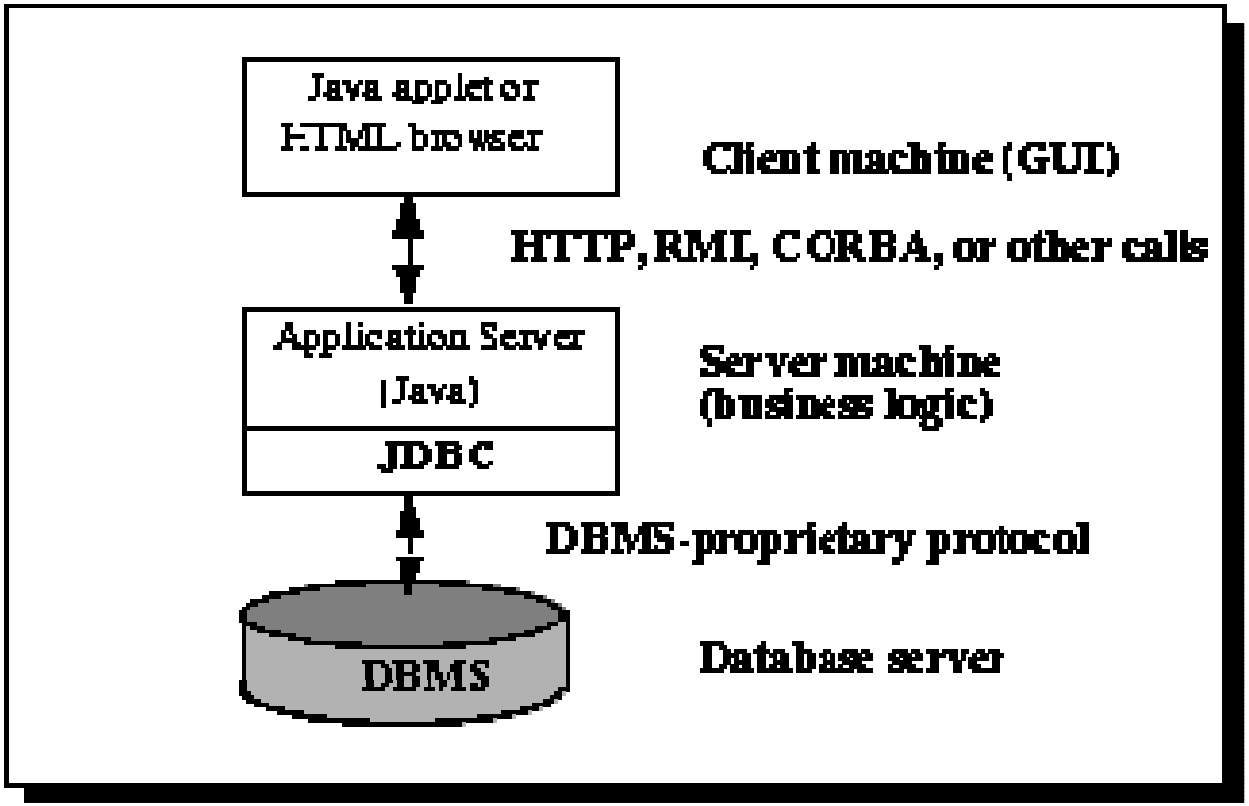
- Kiến trúc 2 lớp



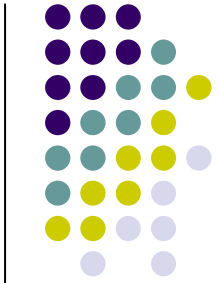
Kiến Trúc Ứng Dụng Sử Dụng JDBC (2)



- Kiến trúc 3 lớp

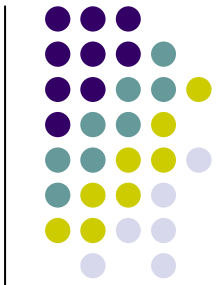


Kết Nối Vào Cơ Sở Dữ Liệu



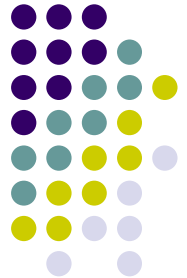
- Để kết nối vào cơ sở dữ liệu cần phải cài đặt JDBC drivers
- Mỗi hệ quản trị cơ sở dữ liệu thường có một số JDBC driver riêng
- Thông tin về các JDBC driver có thể xem ở đây:
  - http://developers.sun.com/product/jdbc/drivers

Sử Dụng Cầu Nối JDBC-OBDC



- Có thể kết nối với cơ sở dữ liệu sử dụng cầu nối JDBC-ODBC
  - ODBC – Open Database Connectivity do Microsoft xây dựng
- Sun có cung cấp sẵn một cầu nối JDBC-ODBC trong Java Development Kit
  - Tên của jdbc-odbc driver của sun: sun.jdbc.odbc.JbdcOdbcDriver

Các Bước Kết Nối



- Load jdbc driver cần sử dụng
  - Class.forName("sun.jdbc.odbc.JbdcOdbcDriver");
- Tạo kết nối (sử dụng cầu nối JDBC-ODBC)
  - Connection conn=DriverManager.getConnection("jdbc:odbc:<DataSourceName>");
  - DataSourceName là tên của ODBC data source được tạo trên MS Windows
  - Các jdbc driver khác nhau sử dụng các chuỗi kết nối khác nhau. Cần đọc hướng dẫn từ nhà cung cấp để biết thông tin này

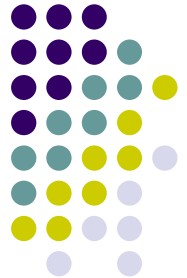
Ví dụ: Kết Nối Với CSDL

```
try{
        Class.forName(driver);
        conn =DriverManager.getConnection(connURL);
}catch(ClassNotFoundException e){
        e.printStackTrace();
        conn = null;
}catch(SQLException sqle){
        sqle.printStackTrace();
        conn = null;
}
```

Thực Thi Các Câu Lệnh SQL – Class Statement

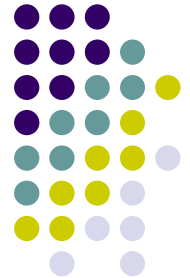
- Sử dụng class Statement để thực thi các câu lệnh SQL trên cơ sở dữ liệu vừa được kết nối
  - Statement st = conn.createStatement();
  - SQLException sẽ được ném ra nếu có lỗi trong quá trình kết nối cơ đến cơ sở dữ liệu
- Phương thức execute(String query) của Statement có thể được dùng để thực thi bất kì câu lện SQL nào

Tạo Bảng Qua JDBC



```
public void createTable(){
    String sqlQuery = "CREATE TABLE PRICELIST(NAME
    VARCHAR(20), PRICE NUMBER);";
    if (conn != null){
        try{
            Statement st = conn.createStatement();
            st.execute(sqlQuery);
            st.close();
        }catch(SQLException e){
            e.printStackTrace();
        }
    }
}
```

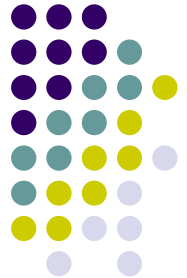
Nhập Dữ Liệu Qua JDBC



- Sử dụng phương thức executeUpdate() của Statement để cập nhật dữ liệu qua JDBC
  - executeUpdate() trả về số dòng bị ảnh hưởng sau khi câu lệnh được thực thi

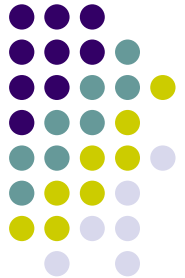
```
public void insertData(String name, double val){
    String insertQuery = "INSERT INTO PRICELIST VALUES('" + name + "'," + val + ");";
    if (conn != null){
        try{
            Statement st = conn.createStatement();
            int d = st.executeUpdate(insertQuery);
            System.out.println("So dong duoc cap nhat: " + d);
            st.close();
        }catch(SQLException e){
            e.printStackTrace();
        }
    }
}
```

Đọc Dữ Liệu Từ Cơ Sở Dữ Liệu



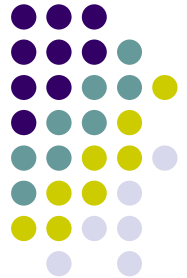
- Sử dụng phương thức executeQuery() của Statement để đọc dữ liệu từ cơ sở dữ liệu
- Dữ liệu đọc được sẽ lưu trong ResultSet
- Có thể di chuyển tới lui trong ResultSet để lấy dữ liệu ra

Ví dụ



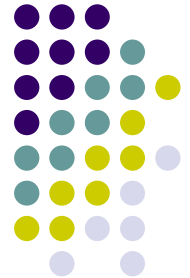
```
public void selectAll(){
    String selectQuery = "SELECT * FROM PRICELIST;";
    if (conn != null){
        try{
            Statement st =
            conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                                 ResultSet.CONCUR_READ_ONLY);
            ResultSet rs = st.executeQuery(selectQuery);
            System.out.println("NAME\t\tPRICE");
            while (rs.next()){
                String name = rs.getString("NAME");
                double val = rs.getDouble("PRICE");
                System.out.println(name + "\t\t" + val);
            }
            st.close();
        }catch(SQLException e){
            e.printStackTrace();
        }
    }
}
```

Các kiểu ResultSet



- TYPE_FORWARD_ONLY
  - Con trỏ của ResultSet kiểu này chỉ được di chuyển theo một hướng từ đầu đến cuối
- TYPE_SCROLL_INSENSITIVE
  - Con trỏ có thể di chuyển tới lui tương đối với vị trí hiện tại của nó, và cũng có thể di chuyển đến một vị trí cụ thể, không bị ảnh hưởng nếu kết quả được thay đổi ở nơi khác
- TYPE_SCROLL_SENSITIVE
  - Con trỏ có thể di chuyển tới lui tương đối với vị trí hiện tại của nó, và cũng có thể di chuyển đến một vị trí cụ thể, sẽ bị ảnh hưởng nếu kết quả bị thay đổi ở nơi khác

Chế Độ Hoạt Động Đồng Thời Của ResultSet

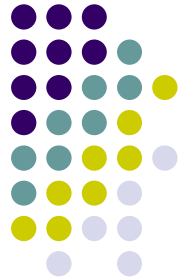


- CONCUR_READ_ONLY
  - Xác định chế độ hoạt động đồng thời, kết quả lưu trong đối tượng ResultSet không được thay đổi
- CONCUR_UPDATABLE
  - Xác định chế độ hoạt động đồng thời, kết quả lưu trong đối tượng ResultSet được thay đổi

Các Phương Thức Của ResultSet

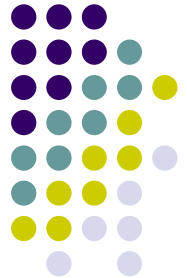
- next(): di chuyển con trỏ đến dòng kế, trả về true nếu có dòng kế tiếp, false nếu đến cuối ResultSet
- previous(): di chuyển con trỏ đến dòng trước
- first(): di chuyển con trỏ đến dòng đầu tiên
- last(): di chuyển con trỏ đến dòng cuối cùng
- beforeFirst(): di chuyển con trỏ đến vị trí trước dòng đầu tiên
- afterLast(): di chuyển con trỏ đến sau dòng cuối cùng
- relative(int rows): di chuyển con trỏ tướng đối với vị trí hiện tại của nó với số dòng là rows
- absolute(int row): di chuyển con trỏ đến dòng thứ row

Lấy Dữ Liệu Từ ResultSet object



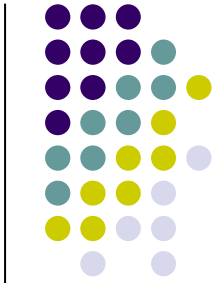
- Dùng phương thức getXXX(String colname)
  - XXX là kiểu dữ liệu được trả về
  - colname là tên của cột cần lấy dữ liệu ra
- Ví dụ:
  - String name = rs.getString("NAME");
  - double val = rs.getDouble("PRICE");

Chỉnh Sửa Dữ Liệu Sử Dụng ResultSet



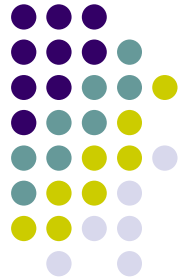
- Dữ liệu của một bảng trong CSDL có thể được chỉnh sửa bằng cách sử dụng câu lệnh SQL UPDATE
- Ta cũng có thể chỉnh sửa dữ liệu trên các hàng của một bảng từ ResultSet
- Để làm được điều này, ResultSet phải được đặt ở chế độ CONCUR_UPDATABLE
- Sử dụng các phương thức updateXXX() để chỉnh sửa dữ liệu trên ResultSet
  - XXX là kiểu dữ liệu của cột cần được sửa

Ví Dụ Chỉnh Sửa Dữ Liệu



```
public void updateTable(){
    String selectQuery = "SELECT * FROM PRICELIST;";
    if (conn != null){
        try{
            Statement st =
            conn.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                 ResultSet.CONCUR_UPDATABLE);
            ResultSet rs = st.executeQuery(selectQuery);
            rs.next();
            rs.updateDouble("PRICE", 1.5);
            rs.updateRow();
            st.close();
        }catch(SQLException e){
            e.printStackTrace();
        }
    }
}
```

Sử Dụng Lớp PreparedStatement

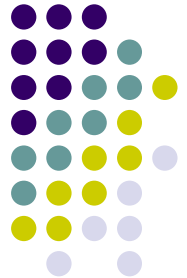


- Có những trường hợp ta cần thực hiện nhiều câu lệnh SQL có cấu trúc tương tự nhau, chỉ có giá trị là thay đổi
- PreparedStatement có thể được sử dụng để soạn trước câu lệnh có sẵn cấu trúc cần thiết
- Giá trị sẽ được đưa vào như những đối số khi câu lệnh được thực thi

Ví dụ: PreparedStatement

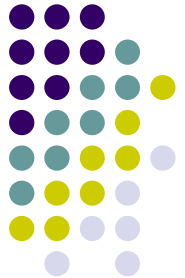
```
public void prepareStatement(){
	String insertQuery = "INSERT INTO PRICELIST VALUES(?,?);";
	if (conn != null){
		try{
			PreparedStatement prest = conn.prepareStatement(insertQuery);
			prest.setString(1, "Biscuit");
			prest.setDouble(2, 1.2);
			prest.executeUpdate();
			prest.setString(1, "Pen");
			prest.setDouble(2, 0.5);
			prest.executeUpdate();
			prest.close();
		}catch(SQLException e){
			e.printStackTrace();
		}
	}
}
```

Sử Dụng Transaction (Giao Dịch)



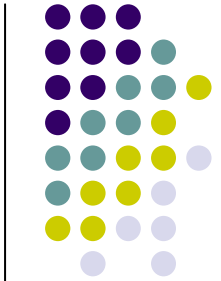
- Mặc định, sau khi mỗi câu lệnh SQL được thực thi qua JDBC, dữ liệu sẽ được cập nhật ngay vào CSDL
- Có những trường hợp, ta muốn dữ liệu chỉ được cập nhật vào CSDL sau khi một số câu lệnh SQL được thực hiện
  - Ví dụ: đối với trang ứng dụng bán hàng qua mạng, để CSDL được thống nhất, ta chỉ muốn lưu các dữ liệu liên quan tới một đơn đặt hàng cùng một lúc
- Một nhóm các câu lệnh như thế được gọi là một giao dich (transaction)

Cách Đặt Giao Dich



- Trước hết, phải không dùng chế độ COMMIT tự động từ Connection object
  - conn.setAutoCommit(false);
- Thực hiện các câu lệnh trong một giao dịch
- Thực hiện COMMIT (lưu) CSDL
  - conn.commit();
- Nếu không cần dùng ở chế độ giao dịch nữa, ta nên trả lại chế độ COMMIT tự động
  - conn.setAutoCommit(true);

Ví dụ: Transaction

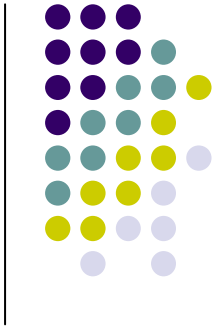


```
public void transaction(){
        String insertQuery = "INSERT INTO PRICELIST VALUES(?,?);";
        if (conn != null){
                try{
                        conn.setAutoCommit(false);
                        PreparedStatement prest = conn.prepareStatement(insertQuery);
                        prest.setString(1, "Biscuit");
                        prest.setDouble(2, 1.2);
                        prest.executeUpdate();
                        prest.setString(1, "Pen");
                        prest.setDouble(2, 0.5);
                        prest.executeUpdate();
                        prest.close();
                        conn.commit();
                        conn.setAutoCommit(true);
                }catch(SQLException e){
                        e.printStackTrace();
                }
        }
}
```

Hủy Một Giao Dịch

- Trong quá trình thực hiện một giao dịch, nếu có sai sót nào xảy ra, ta có thể hủy giao dịch đang được thực hiện nửa chừng bằng cách sử dụng rollback
- conn.rollback() sẽ hũy dữ liệu đến thời điểm nó được commit gần nhất
  - Tùy thuộc vào hệ quản trị CSDL mà cách rollback có thể khác nhau
- Dữ liệu sau khi được commit sẽ không hủy được bằng rollback

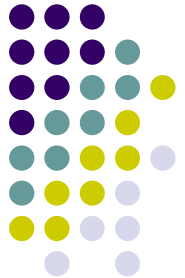
Ví Dụ: rollback



```
public void rollbackTransaction(){
        String insertQuery = "INSERT INTO PRICELIST VALUES(?,?);";
        if (conn != null){
                try{
                        conn.setAutoCommit(false);
                        PreparedStatement prest = conn.prepareStatement(insertQuery);
                        prest.setString(1, "Biscuit");
                        prest.setDouble(2, 1.2);
                        prest.executeUpdate();
                        conn.commit();
                        prest.setString(1, "Pen");
                        prest.setDouble(2, 0.5);
                        prest.executeUpdate();
                        prest.close();
                        conn.rollback();
                        conn.commit();
                        conn.setAutoCommit(true);
                }catch(SQLException e){
                        e.printStackTrace();
                }
        }
}
```

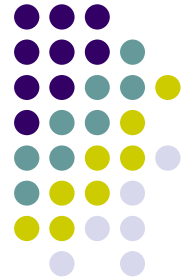
Thông tin về Pen sẽ không được lưu vào CSDL

ResultSetMetaData



- Để lấy thông tin (Metadata) về một ResultSet nào đấy

Ví dụ: ResultSetMetaData



```
public void showMetadata(){
    String selectQuery = "SELECT * FROM PRICELIST;";
    if (conn != null){
        try{
            Statement st = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                                                ResultSet.CONCUR_READ_ONLY);
            ResultSet rs = st.executeQuery(selectQuery);
            ResultSetMetaData meta = rs.getMetaData();
            int iColumnCount = meta.getColumnCount();
            for (int i =1 ; i <= iColumnCount ; i++){
                System.out.println("Column Name: " + meta.getColumnName(i));
                System.out.println("Column Type: " + meta.getColumnTypeName(i));
                System.out.println("Display Size: " + meta.getColumnDisplaySize(i) );
                System.out.println("Precision: " + meta.getPrecision(i));
                System.out.println("Scale: " + meta.getScale(i) );
            }
            st.close();
        }catch(SQLException e){
            e.printStackTrace();
        }
    }
}
```

Kết Thúc