

Bài 6: JDBC



Nội dung

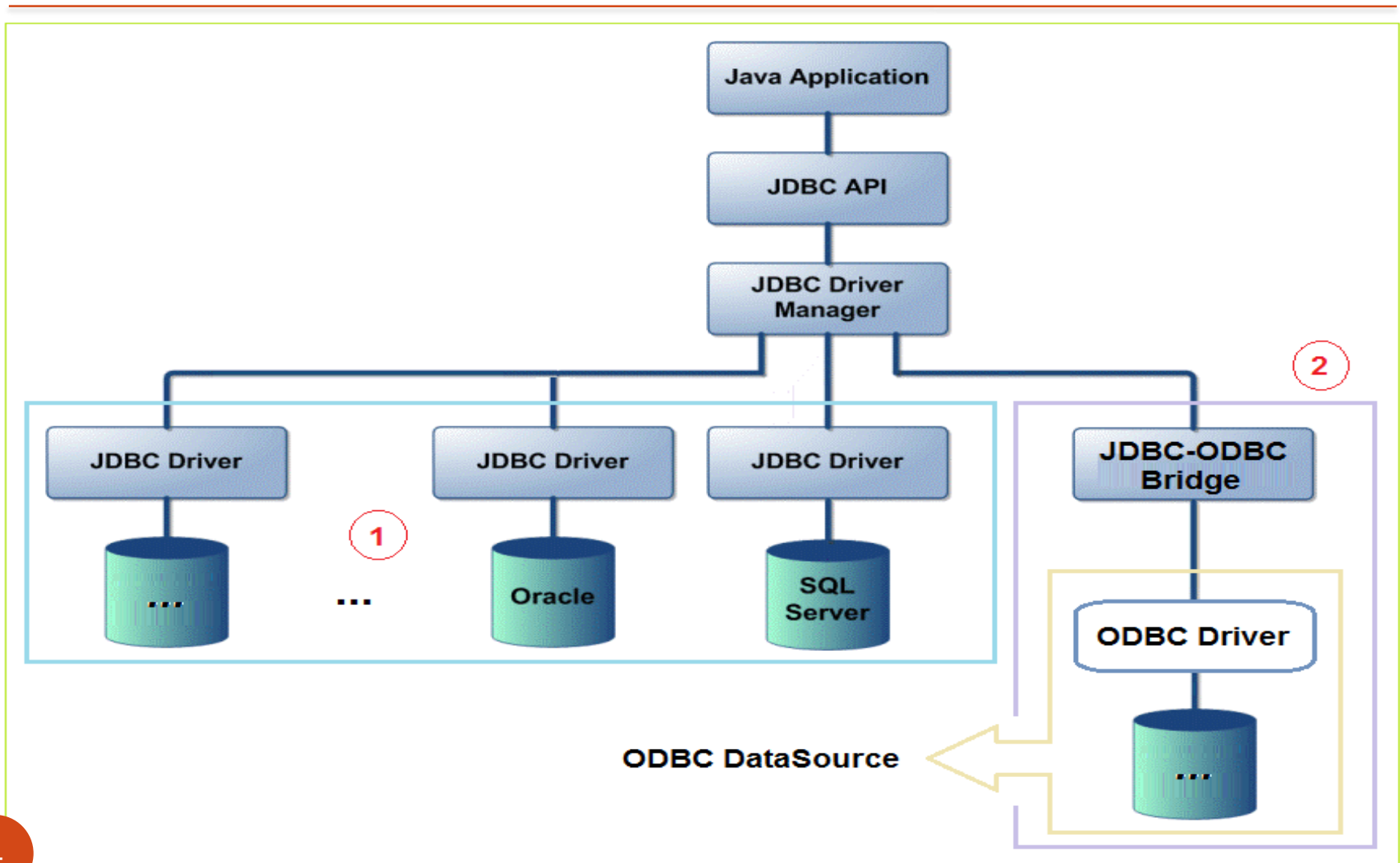
- Giới thiệu JDBC (**JAVA DATABASE**
- JDBC Drivers

Khái niệm



- JDBC hỗ trợ việc truy cập CSDL để thực hiện các tác vụ xử lý (truy vấn, thêm, xóa, sửa, cập nhật)
 - Tạo kết nối đến Database
 - Tạo câu lệnh truy vấn SQL
 - Thực thi các câu lệnh SQL
 - Truy vấn, hiển thị và xử lý dữ liệu trả về
- JDBC có 2 gói hỗ trợ khi lập trình CSDL: `java.sql.*` và `javax.sql.*`

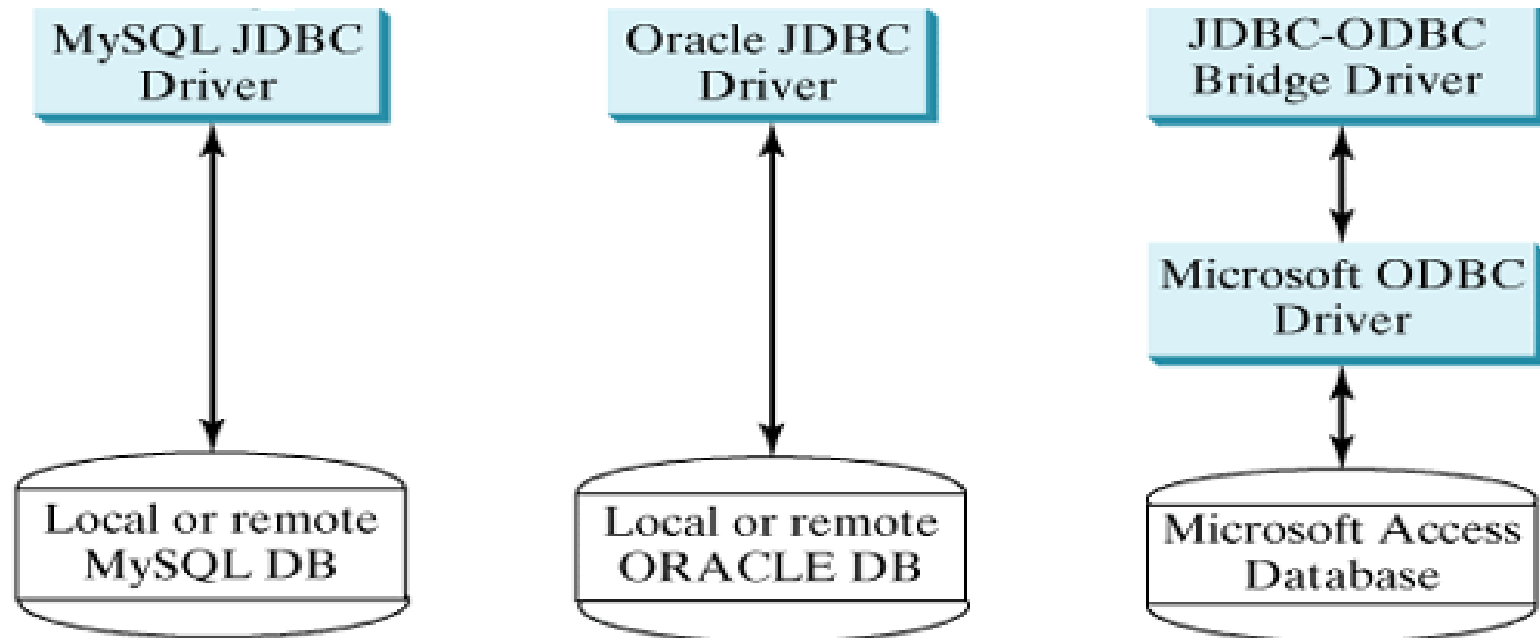
Kiến trúc JDBC



Kiến trúc JDBC (tt)

- **JDBC API**: Tập hợp các interface cung cấp trực tiếp cho ứng dụng để thao tác với JDBC
- **DriverManager**: quản lý các trình điều khiển JDBC (JDBC Driver) được coi là xương sống của JDBC
- **JDBC Driver**: là thành phần chính giao tiếp trực tiếp với CSDL

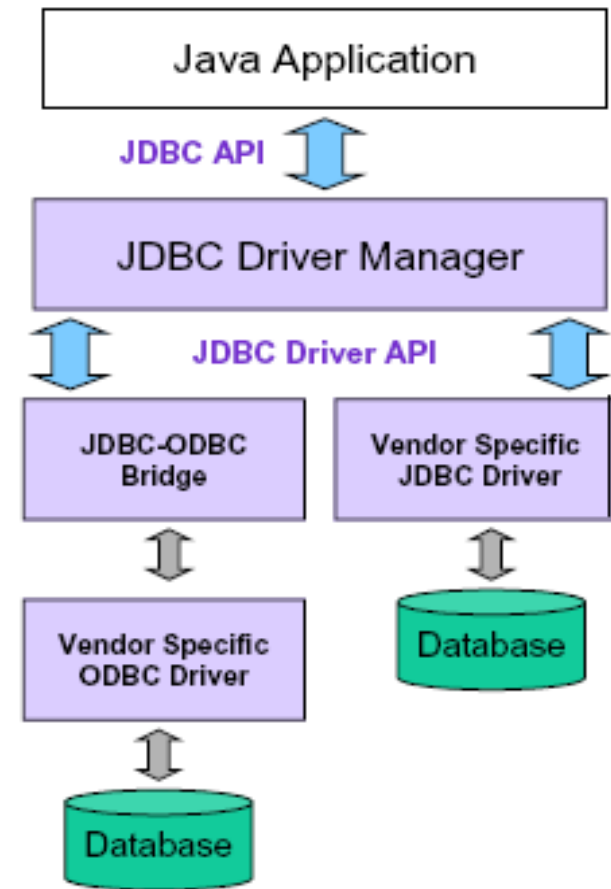
Kiến trúc JDBC - JDBC Driver



JDBC DRIVERS



- JDBC bao gồm 02 thành phần
 - JDBC API
 - JDBC Driver Manager
- JDBC sử dụng 04 loại Driver
 - **Loại 1**: JDBC ODBC
 - **Loại 2**: Native API
 - **Loại 3**: Network Protocol
 - **Loại 4**: Native Protocol



Loại 1: JDBC ODBC



- Các trình điều khiển cầu nối JDBC-ODBC.

- Ủy nhiệm công việc truy xuất dữ liệu cho ODBC API

- SUN cung cấp một phần mềm trình điều khiển JDBC/ODBC.

- **Ưu điểm:**

- Dễ kết nối, kết nối thẳng đến CSDL

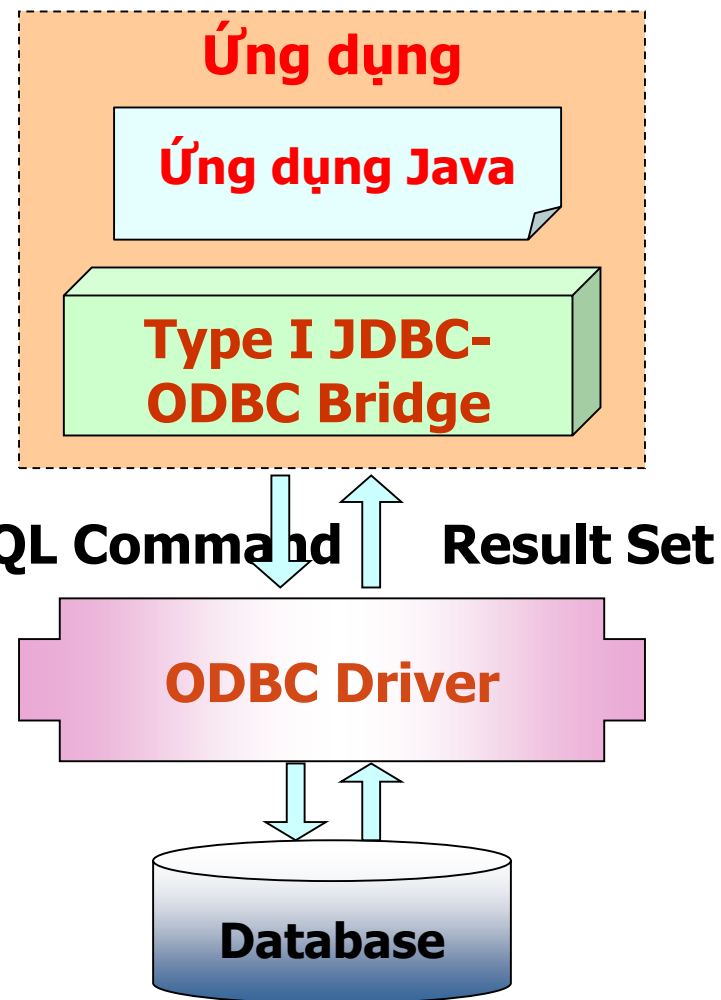
- Áp dụng khi không có driver của **SQL Command** **Result Set** CSDL

- **Nhược điểm:**

- Lệ thuộc ODBC driver

- Thời gian thực hiện chậm

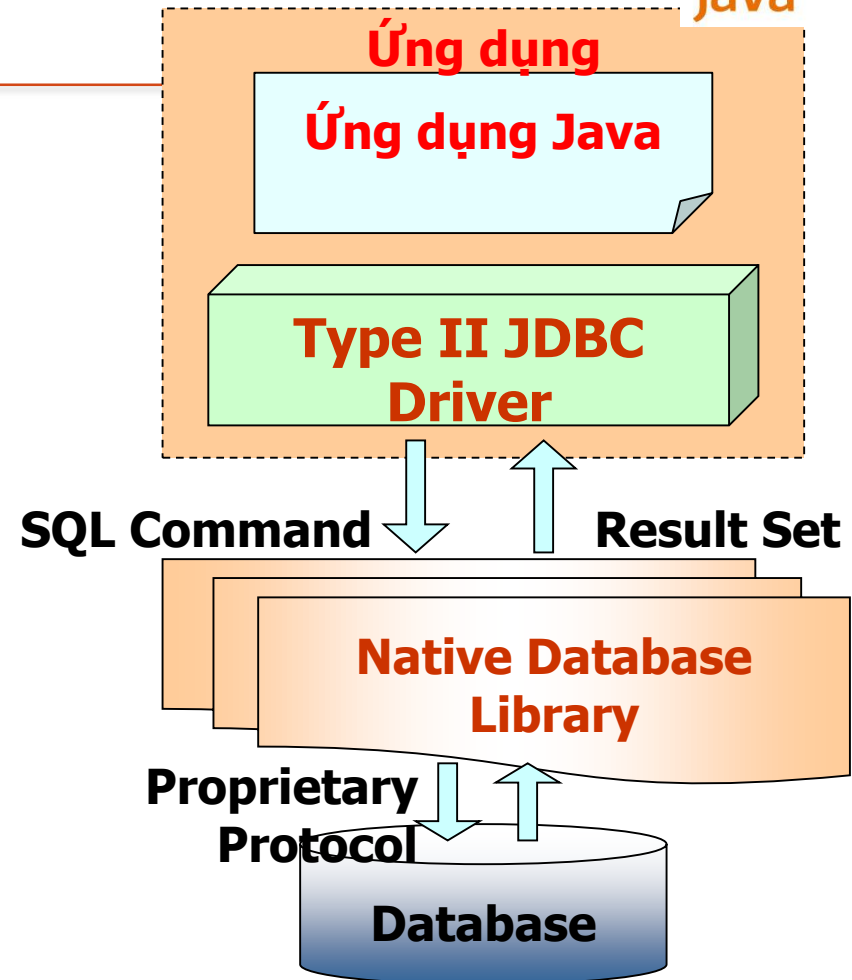
- Client khai thác CSDL và ODBC phải cùng một server



Loại 2: NATIVE API



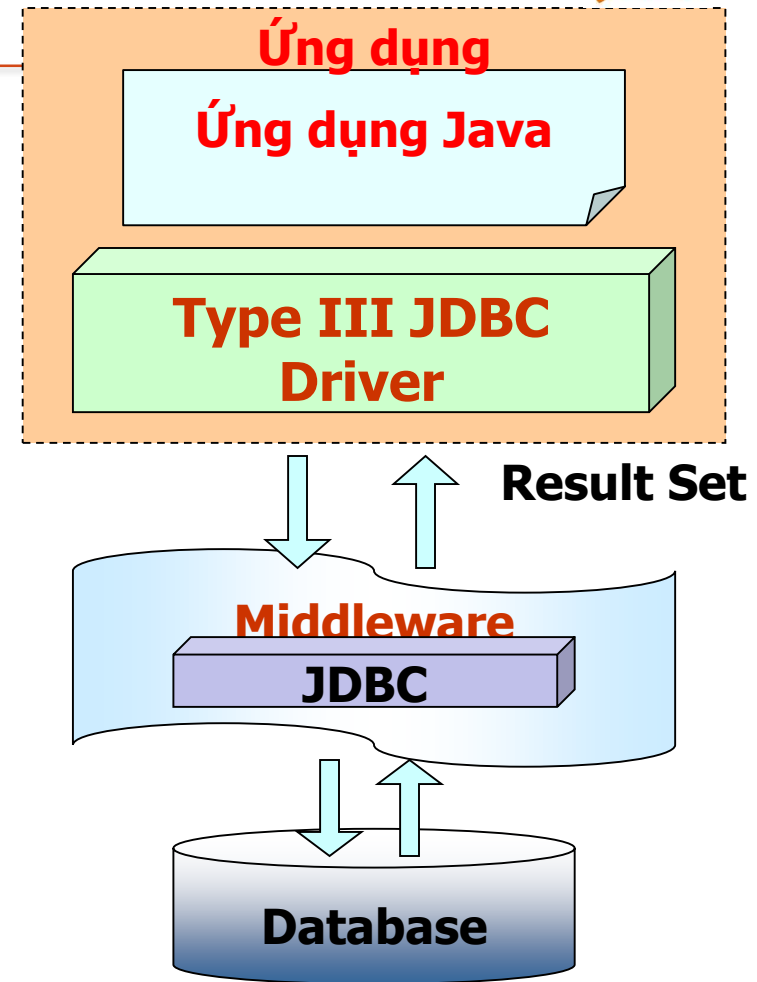
- Tương tự type 1, thay thế ODBC bằng native code
- Chuyển lệnh gọi JDBC thành native C/C++ API, tùy thuộc vào database
- Được cung cấp bởi database vendor.
- Driver của CSDL cần cài đặt trên mỗi máy chạy ứng dụng
- Nâng cao hiệu quả thực hiện kết nối hơn type 1



Loại 3: NETWORK PROTOCOL



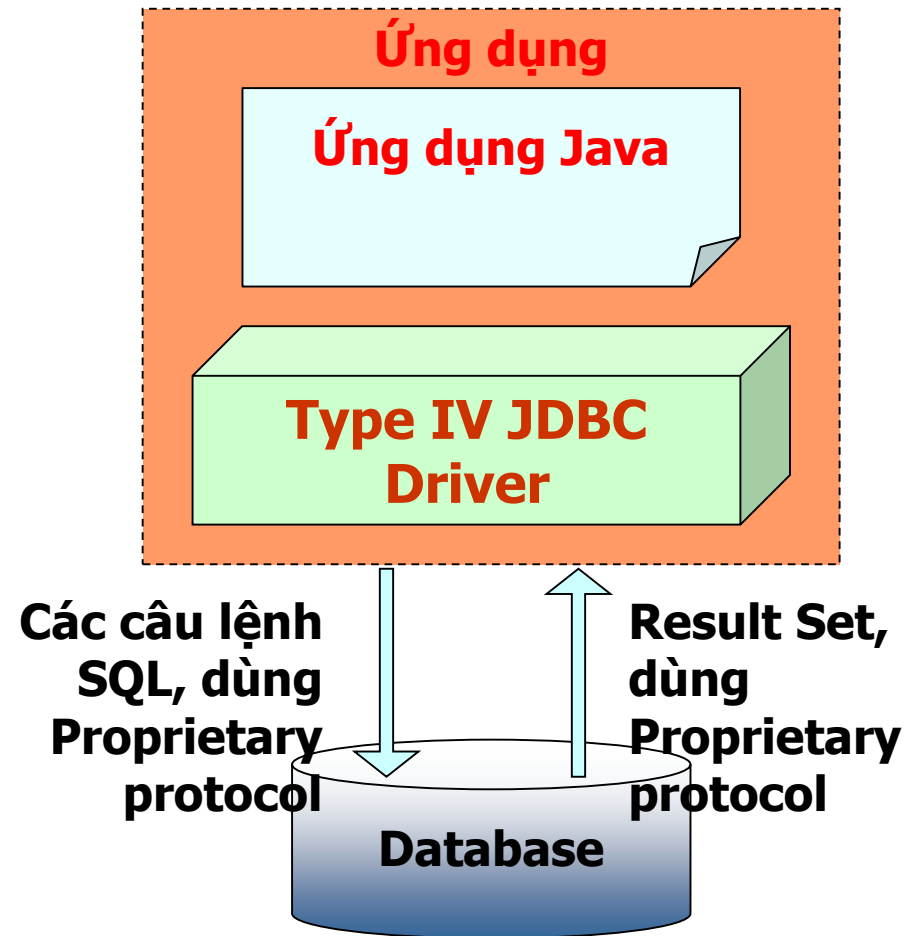
- Được viết thuần bằng Java và sử dụng giao thức Net độc lập nhà sản xuất để truy cập đến trình theo dõi từ xa.
- Truy cập CSDL qua thành phần middle ware
- Hỗ trợ việc kết nối nhiều ứng dụng với nhiều CSDL khác nhau
- **Ưu điểm:** có thể kết nối đến nhiều hệ quản trị CSDL khác nhau mà không cần cài đặt driver trên client
- **Nhược điểm:** phụ thuộc vào nhà cung cấp phần mềm trung gian



Loại 4: NATIVE PROTOCOL



- Được viết thuần túy bằng Java, là loại hiệu quả nhất.
- Kết nối trực tiếp vào CSDL
- Các drivers được hỗ trợ bởi các provider DBMS
- **Ưu điểm:**
 - Nâng cao hiệu quả khi thực thi
 - Độc lập platform
- **Nhược điểm :** đòi hỏi có driver cho từng loại CSDL



Nên dùng loại nào???



- Nếu ứng dụng truy cập vào một loại cơ sở dữ liệu như MySQL, SQL Server, PostgreSQL, Oracle, ... thì **loại 4** nên được sử dụng.
- Nếu truy cập nhiều loại cơ sở dữ liệu cùng một lúc thì **loại 3** nên được sử dụng.
- Khi loại 3 hoặc loại 4 không có sẵn cho cơ sở dữ liệu thì loại 2 được sử dụng trong một số tình huống.
- Loại 1 thường được sử dụng với mục đích thử nghiệm.

JDBC Interface



- Class và Interface của JDBC API thuộc gói java.sql
- **DriverManager**: dùng để nạp các driver và tạo Connection đến cơ sở dữ liệu.
- **Driver**: Driver của cơ sở dữ liệu, mỗi JDBC Driver đều cài đặt lại Interface này.
- **Connection** : thiết lập một Connection đến cơ sở dữ liệu và cho phép tạo các Statement .
- **Statement**: gắn kết với một connection đến cơ sở dữ liệu và cho phép thực thi các câu lệnh SQL.
- **PreparedStatement**: tương tự như Statement nhưng thực thi câu lệnh SQL được biên dịch trước (Precompiled SQL) và có truyền tham số

JDBC Interface (tt)



- **ResultSet**: Cung cấp thông tin rút trích từ cơ sở dữ liệu, cho phép truy xuất các dòng dữ liệu.
- **ResultSetMetaData**: Cung cấp các thông tin như kiểu dữ liệu và các thuộc tính trong Resultset.
- **DatabaseMetaData**: Cung cấp các thông tin của cơ sở dữ liệu kết nối.
- **SQLException**: Cung cấp thông tin các ngoại lệ xảy ra khi tương tác với cơ sở dữ liệu.

CÁC BƯỚC SỬ DỤNG JDBC



- Đăng ký **driver** của JDBC (Load driver)
- Xác định các thông số CSDL – DB kết nối
- Tạo kết nối **CSDL**
- Tạo lệnh **SQL** cần thực thi
- Thực thi lệnh
- Xử lý kết quả trả về
- Đóng **Connection**

Đăng ký driver (LOAD DRIVER)



- Driver là phần mềm hỗ trợ giao tiếp.
- JDBC driver là Java class thực hiện chuyển đổi các lệnh Java thành câu lệnh SQL tương ứng.
- Load driver là tạo các instance hỗ trợ liên kết và đăng ký với JDBC
- Load Driver:
 - `Class.forName("JDBCDriverClass");`
- Kiểm tra DriverClass có tồn tại hay không:
 - `ClassNotFoundException`

Đăng ký driver (LOAD DRIVER) (tt)



Trình quản trị CSDL	JDBCDriverClass
Access	sun.jdbc.odbc.JdbcOdbcDriver
MySQL	com.mysql.jdbc.Driver
SQL Server	com.microsoft.sqlserver.jdbc.SQLServerDriver
Oracle	oracle.jdbc.driver.OracleDriver

➤ VD khi dùng MySQL

```
Class.forName(com.mysql.jdbc.Driver)
```

➤ VD khi dùng SQL Server

```
Class.forName(com.microsoft.sqlserver.jdbc.SQLServerDriver)
```

Đăng ký driver (LOAD DRIVER) (tt)



```
try {  
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver")  
    Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");  
}  
catch (ClassNotFoundException e) {  
    System.out.println("Error loading driver: " + e);  
}
```

Xác định các thông số kết nối



- Thông số kết nối CSDL sẽ thay đổi tùy theo loại **driver** kết nối.
- **ODBC:**
 - jdbc:odbc: **tênDSN**
 - **Ví dụ:** jdbc:odbc:NWind
- **Khác:**
 - Xác định provider, port, database liên kết, ...
 - **Ví dụ:**
 - jdbc:mysql://localhost:1433/pubs
 - jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=pubs

Tạo kết nối đến CSDL



```
Connection con =  
DriverManager.getConnection(databaseURL,  
username,password);
```

Hệ quản trị CSDL	databaseURL
Access	jdbc:odbc:dataSource
MySQL	jdbc:mysql://hostname/dbname
SQL Server	jdbc:Microsoft: sqlserver://hostname/dbname
Oracle	jdbc:oracle:thin: @hostname:port#:oracleDBSID

Ví dụ Load Driver và tạo kết nối



➤ MySQL

```
String url =  
"jdbc:mysql://localhost:3306/world?user=root&password=&u  
seUnicode=true&characterEncoding=utf8";  
String driver = "com.mysql.jdbc.Driver";  
try{  
    Class.forName(driver);  
    con = DriverManager.getConnection(url);  
    System.out.println("Connected to the database");  
    return true;  
} catch (Exception e) {  
    e.printStackTrace();  
    return false;  
}
```

Tạo kết nối SQL Server



➤ Tạo kết nối trong SQL Server

```
String userName = "hung";  
String password = "12345";  
Class.forName("com.microsoft.sqlserver.jdbc  
    .SQLServerDriver");  
String url =  
    "jdbc:sqlserver://localhost:1433;databaseN  
ame=qlhs;";  
Connection con =  
    java.sql.DriverManager.getConnection(url, u  
serName, password);
```

Tạo câu lệnh thực thi



- `Statement statement = connection.createStatement() ;`
- `ResultSet executeQuery (String sql)`
- `int executeUpdate (String sql)`
- `boolean execute (String sql)`

Tạo câu lệnh thực thi



```
String SQL = "SELECT * FROM hocsinh";
```

```
Statement stat = con.createStatement();
```

○ Có 03 loại Statement

- CreateStatement

- PreparedStatement (prepareStatement).

```
Select * From Registration Where uName = ?
```

- CallableStatement (prepareCall()).

Ví dụ: {stpInsert (?) }

Thực thi lệnh



- `executeQuery()` đối với câu lệnh truy vấn

```
String strSQL = "Select * From Registration";  
ResultSet rs = stat.executeQuery(strSQL);
```

- `executeUpdate()`: đối câu lệnh **Insert**, **Update** và **Delete**

```
String strSQL = "Insert into Registration  
Values ("Aptech", "Aptech");  
int nRow = stat.executeUpdate(strSQL);
```

- `execute()` dùng để tạo và xóa đối tượng như **table**

```
String strSQL = "Drop table Registration";  
stat.execute(strSQL);
```

Xử lý kết quả trả về và đóng kết nối



➤ Xử lý kết quả

- Sử dụng **ResultSet** nhận kết quả trả về
- Sử dụng phương thức **getXxx** (số thứ tự/ hay tên field) của **ResultSet** để lấy giá trị của **field**
 - Số thứ tự bắt đầu từ 1
 - Xxx tương ứng với loại dữ liệu của field
 - **getInt()**: lấy về giá trị **int** từ dòng hiện hành.
 - **getString()**: lấy về giá trị **String** từ dòng hiện hành.
 - **getDate()**: lấy về giá trị **Date** từ dòng hiện hành.
 - **getFloat()**: lấy về giá trị **float** từ dòng hiện hành.
 - **getObject()**: lấy về giá trị từ dòng hiện hành và xem giá trị này như là 1 **object**.
- Sử dụng phương thức **next()** của **ResultSet** để duyệt lần lượt các record

Xử lý kết quả trả về và đóng kết nối



➤ Ví dụ:

```
while (rs.next())  
{  
    rs.getInt(1) hay rs.getInt("userId");  
    rs.getString(2) hay  
    rs.getString("username");  
}
```

- Resultset **2 chiều** hỗ trợ các phương thức truy cập như:
first, isFirst, last, ..

➤ Đóng các kết nối:

- Dùng phương thức close(). **con.close();**

Select



➤ Chọn dòng trong CSDL

```
String SQL = "SELECT * FROM hocsinh";  
Statement stat = conn.createStatement();  
ResultSet rs = stat.executeQuery(SQL);  
int count=0;  
while (rs.next()) {  
    String ten=rs.getString(1);  
    System.out.println(ten);  
    count++;  
}
```

Insert



➤ Chèn một dòng vào một bảng trong CSDL

```
String SQL1="INSERT hocsinh VALUES ('Mai Xuan Den')";  
  
Statement stat1 = conn.createStatement();  
  
stat1.executeUpdate(SQL1);
```

Delete một dòng



➤ Xóa một dòng thỏa mãn điều kiện trong bảng

```
Statement st = con.createStatement();
String sql = "DELETE FROM hocsinh WHERE ten = 'Mai
Xuan Hung'";
int delete = st.executeUpdate(sql);
if(delete >= 1) {
    System.out.println("Row is deleted.");
}
else{
    System.out.println("Row is not deleted.");
}
```

Delete tất cả các dòng

```
Statement st = con.createStatement();  
String sql = "DELETE FROM hocsinh";  
int delete = st.executeUpdate(sql);  
if(delete >= 1){  
    System.out.println("All rows are completely deleted!");  
}
```

Delete một bảng

➤ Xóa một bảng trong CSDL

```
Statement st = con.createStatement();  
st.execute("DROP TABLE Employee1");  
System.out.println ("Table Deletion process is  
completely successfully!");
```


➤ Cập nhật dòng thỏa điều kiện dùng PreparedStatement

```
String sql = "UPDATE movies SET title = ? WHERE  
year_made = ?";  
PreparedStatement prest =  
con.prepareStatement(sql);  
prest.setString(1, "Sanam We wafafa");  
prest.setInt(2, 2005);  
prest.executeUpdate();  
System.out.println("Updating Successfully!");
```

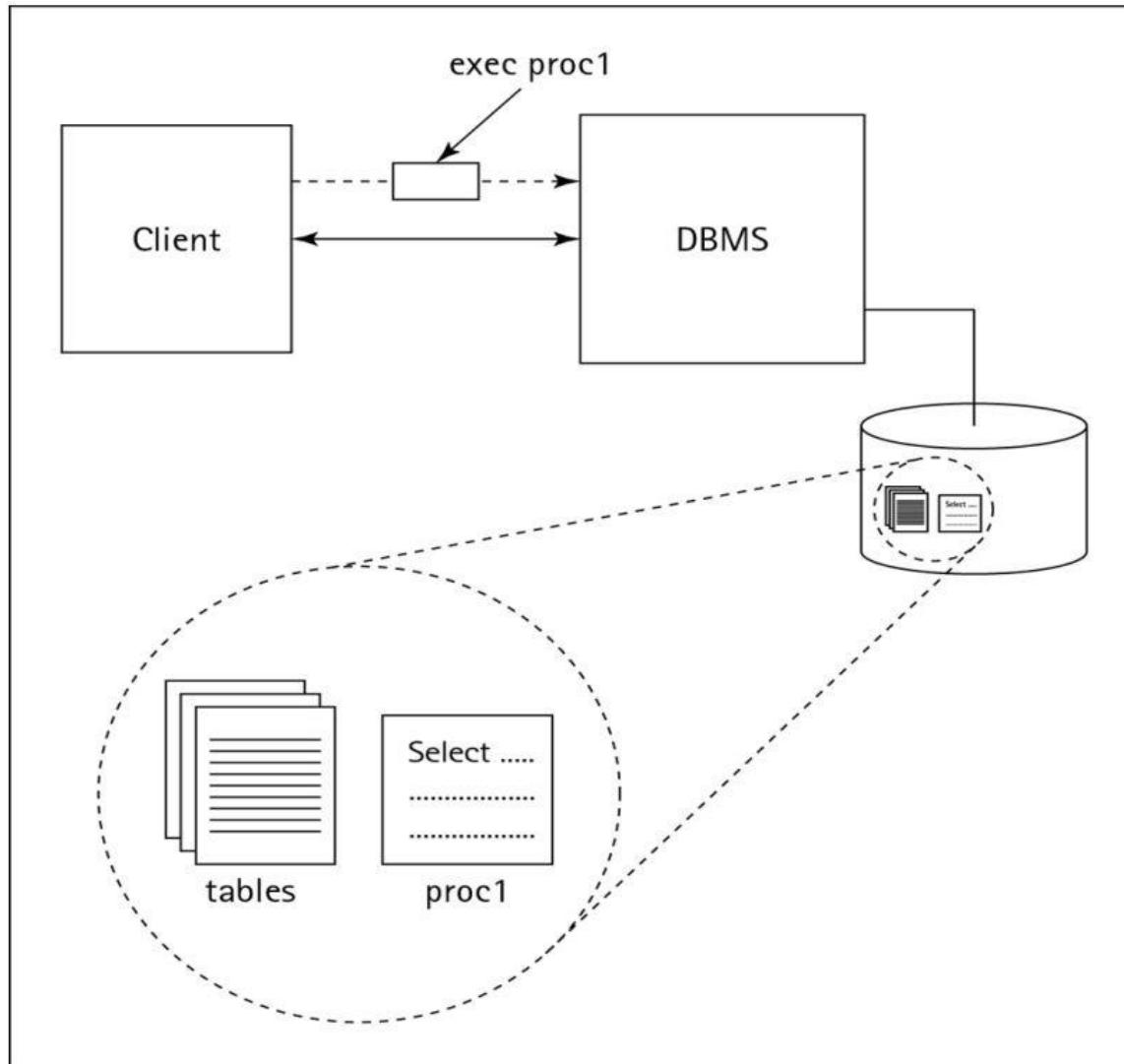
Count Rows



➤ Đếm số dòng dữ liệu trong bảng

```
Statement st = con.createStatement();
BufferedReader bf = new BufferedReader(new
InputStreamReader(System.in));
System.out.println("Enter table name:");
String table = bf.readLine();
ResultSet res = st.executeQuery("SELECT COUNT(*)
FROM "+table);
int count=0;
while (res.next()) {
count ++; }
System.out.println("Number of column:"+count);
```

CallableStatement



CallableStatement



- Gọi **Stored procedure** trong cơ sở dữ liệu
`{call <tên_sp> (? , ? , ? , ...) }`
(Trong đó tương ứng với mỗi dấu ? là 1 tham số của store procedure đã tạo trong Database)
- Tạo 1 thể hiện của interface **CallableStatement** thông qua phương thức **prepareCall()** dựa trên đối tượng **Connection**
- Gọi phương thức **executeQuery()** để trả về kết quả là 1 **ResultSet**.

```
CREATE PROCEDURE getAccounts  
AS  
BEGIN  
SELECT * FROM Test  
END
```

```
String strCall = "{call  
getAccounts}";  
CallableStatement caSt =  
con.prepareCall(strCall);  
ResultSet rs =  
caSt.executeQuery();
```

CallableStatement (tt)



```
CREATE PROCEDURE deleteAccount
@username VARCHAR(50)
AS
BEGIN
    DELETE FROM Logon WHERE [UserName]=@username
END
```

```
String strCall = "{call deleteAccount(?)}";
CallableStatement caSt = con.prepareCall(strCall);
caSt.setString(1, user);
caSt.execute();
```

Sử dụng Transaction



```
try{
    con.setAutoCommit(false);
    Statement statement1= con.createStatement();
    Statement statement2= con.createStatement();
    statement1.executeUpdate(sql1);
    statement2.executeUpdate(sql2);
    con.commit();
}catch (SQLException ex){
    con.rollback();
}finally{
    con.close();
}
```

Sử dụng Transaction (tt)

```
String sql1= insert/delete/update . . .
String sql2= insert/delete/update . . .
try{
    con.setAutoCommit(false) ;
    Statement statement1= con.createStatement();
    Statement statement2= con.createStatement();
    statement1.executeUpdate(sql1);
    statement2.executeUpdate(sql2);
    con.commit() ;
}catch (SQLException ex){
    con.rollback() ;
}finally{
    con.close();
}
```

Sử dụng JTable



- Thuộc gói javax.swing
- Được dùng để hiển thị dữ liệu lên Frame theo dạng bảng biểu
- Cho phép sửa đổi thông tin, đặt kích cỡ và đầu đề cho từng cột
- Các đối tượng hỗ trợ
 - JScrollPane: tạo thanh cuộn trong JTable
 - TableModel: cung cấp dữ liệu hiển thị trên Jtable
 - AbstractTableModel
 - DefaultTableModel

Sử dụng JTable (tt)

➤ Thay đổi kích thước của Jtable

➤ Cú pháp:

```
table.setAutoResizeMode(JTable.AUTO_RESIZE_OFF )
```

- AUTO_RESIZE_ALL_COLUMNS
- AUTO_RESIZE_LAST_COLUMN
- AUTO_RESIZE_NEXT_COLUMN
- AUTO_RESIZE_OFF
- AUTO_RESIZE_SUBSEQUENT_COLUMNS

Lớp TableModel

- Là một interface
- Cung cấp dữ liệu hiển thị cho Jtable
- Cung cấp một số phương thức
 - Số lượng dòng và số lượng cột trong bảng
 - Kiểu dữ liệu trong từng cột
 - Tiêu đề cho từng cột
 - Có cho phép sửa giá trị trong một ô hay không

TableModel
<ul style="list-style-type: none">◆ <code>getRowCount() : int</code>◆ <code>getColumnCount() : int</code>◆ <code>getValueAt(row : int, column : int) : Object</code>◆ <code>getColumnName(column : int) : String</code>◆ <code>getColumnClass(column : int) : Class</code>◆ <code>isCellEditable(row : int, column : int) : boolean</code>◆ <code>addTableModelListener(listener : TableModelListener) : void</code>◆ <code>removeTableModelListener(listener : TableModelListener) : void</code>

Sử dụng DefaultTableModel

➤ Khai báo:

```
private JTable jTable = new JTable();
private DefaultTableModel tableModel = new
DefaultTableModel();
String []colsName = {"Mã nước","Tên nước"};
tableModel.setColumnIdentifiers(colsName);
jTable.setModel(tableModel);
//Code truy vấn dữ liệu ở đây và đưa vào result
while (result.next()){
    String rows[] = new String[2];
    rows[0] = result.getString(1);
    rows[1] = result.getString(2);
    tableModel.addRow(rows);
}
```

Sử dụng AbstractTableModel



- Xây dựng một lớp kế thừa AbstractTableModel
- Cài đặt bắt buộc 3 phương thức:
 - Phương thức trả về số dòng của bảng: `int getRowCount()`
 - Phương thức trả về số cột của bảng: `int getColumnCount()`
 - Phương thức trả về giá trị của mỗi ô: `Object getValueAt(int row, int column)`
- Dữ liệu được lưu trong một bảng Object 2 chiều hoặc kiểu List chứa đối tượng Object

```
Object[][] DATA;  
List<Object[]> DATA = new ArrayList<Object[]>();
```

Sử dụng AbstractTableModel (tt)



➤ Một số phương thức khác:

○ Phương thức lấy tên cột:

- `String getColumnName(int column)`
- Tên cột sử dụng kiểu dữ liệu mảng

```
String[] COLUMN_NAMES = {"Col 01", "Col 02", "Col 03"};
```

○ Phương thức gán giá trị vào một vị trí

- `setValueAt(Object value, int row, int column)`

Q & A

