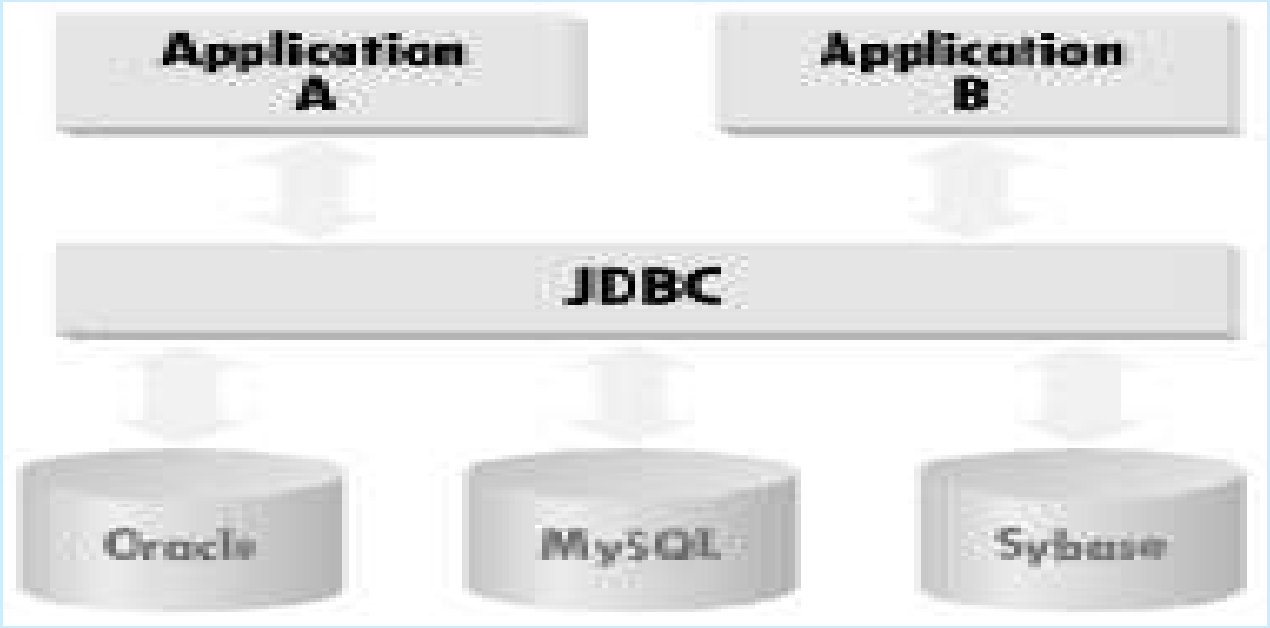
Chương 7 :
LẬP TRÌNH CƠ SỞ DỮ LiỆU

KẾT NỐI CSDL

Java Database Connectivity

Tổng Quan

- JDBC cung cấp tập các lớp và interface cho phép chương trình Java có thể làm việc được với hệ CSDL
- Tập các lớp của JDBC có thể làm việc được với mọi hệ csdl.



Database Driver

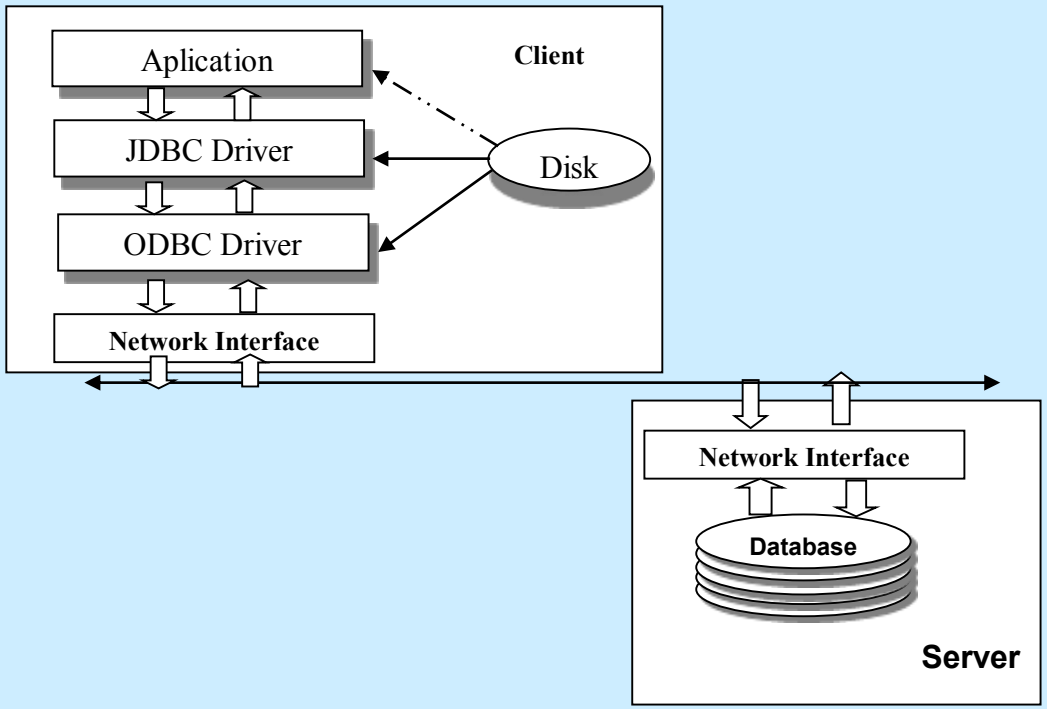
- Bảo đảm ứng dụng java tương tác với mọi csdl dưới một cách thức chuẩn và duy nhất.
- Bảo đảm những yêu cầu từ chương trình sẽ được biểu diễn trong csdl dưới một ngôn ngữ mà csdl hiểu được
- Nhận các yêu cầu từ client, chuyển nó vào định dạng mà csdl có thể hiểu được và thể hiện trong csdl.
- Nhận các phản hồi, chuyển nó ngược lại định dạng dữ liệu java và thể hiện trong ứng dụng.

JDBC Driver

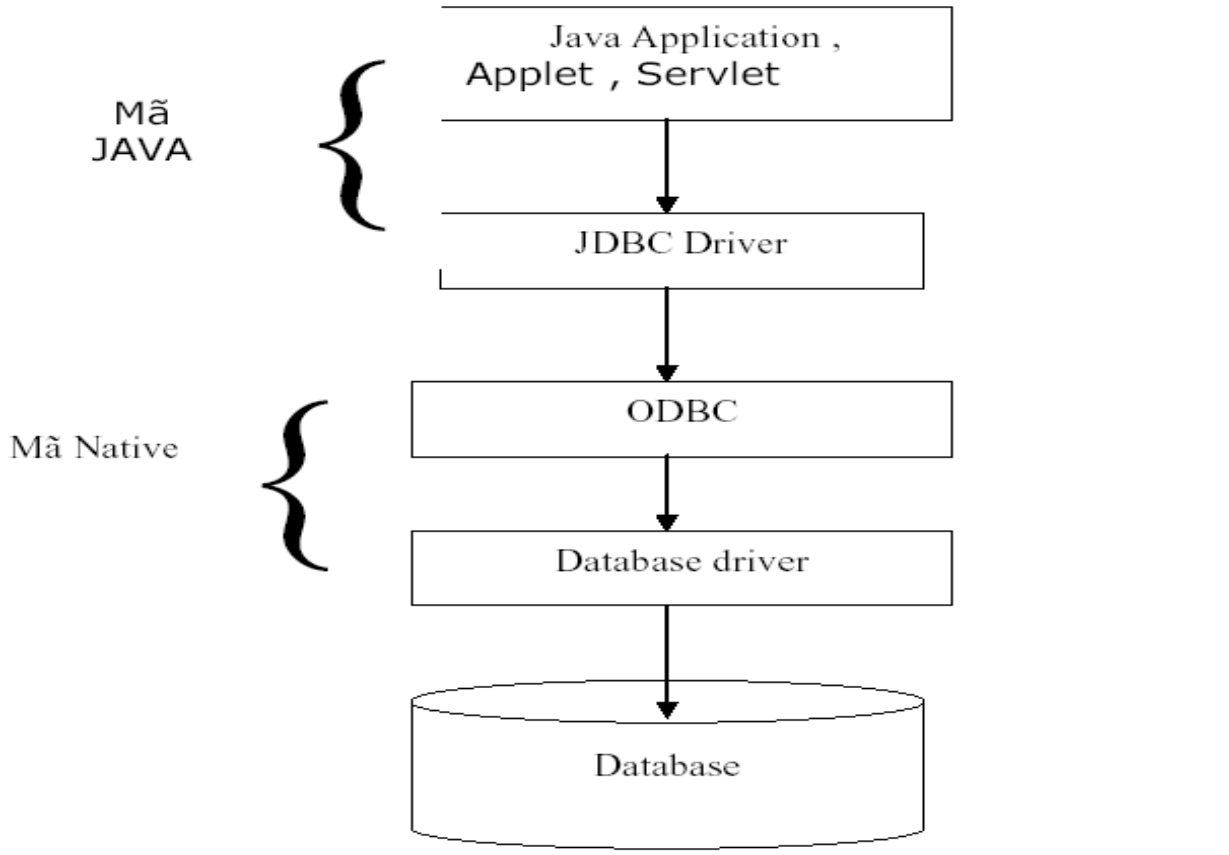
- Có 4 loại JDBC Driver
  - Loại 1 : JDBC sử dụng cầu nối ODBC
  - Loại 2 : JDBC kết nối trực tiếp với các trình điều khiển
  - Loại 3 : JDBC kết nối thông qua các ứng dụng mạng trung gian
  - Loại 4 : JDBC kết nối thông qua các trình điều khiển đặc thù ở xa
- Loại 2,3,4 nói chung được viết bởi nhà cung cấp csdl, hiệu quả hơn loại 1 nhưng thực hiện phức tạp hơn.

JDBC SỬ DỤNG CẦU NỐI ODBC

- jdk hỗ trợ cầu nối jdbc-odbc (jdbc-odbc bridge).
- Mềm dẻo nhưng không hiệu quả.



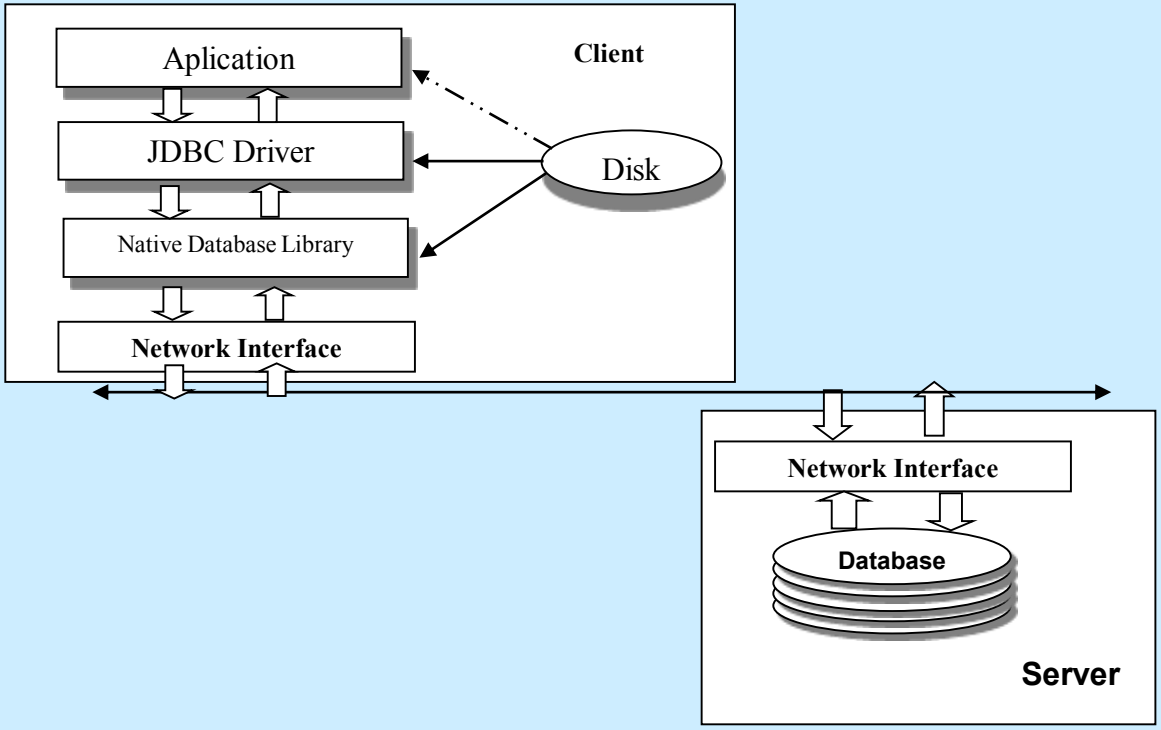
JDBC sử dụng cầu nối ODBC



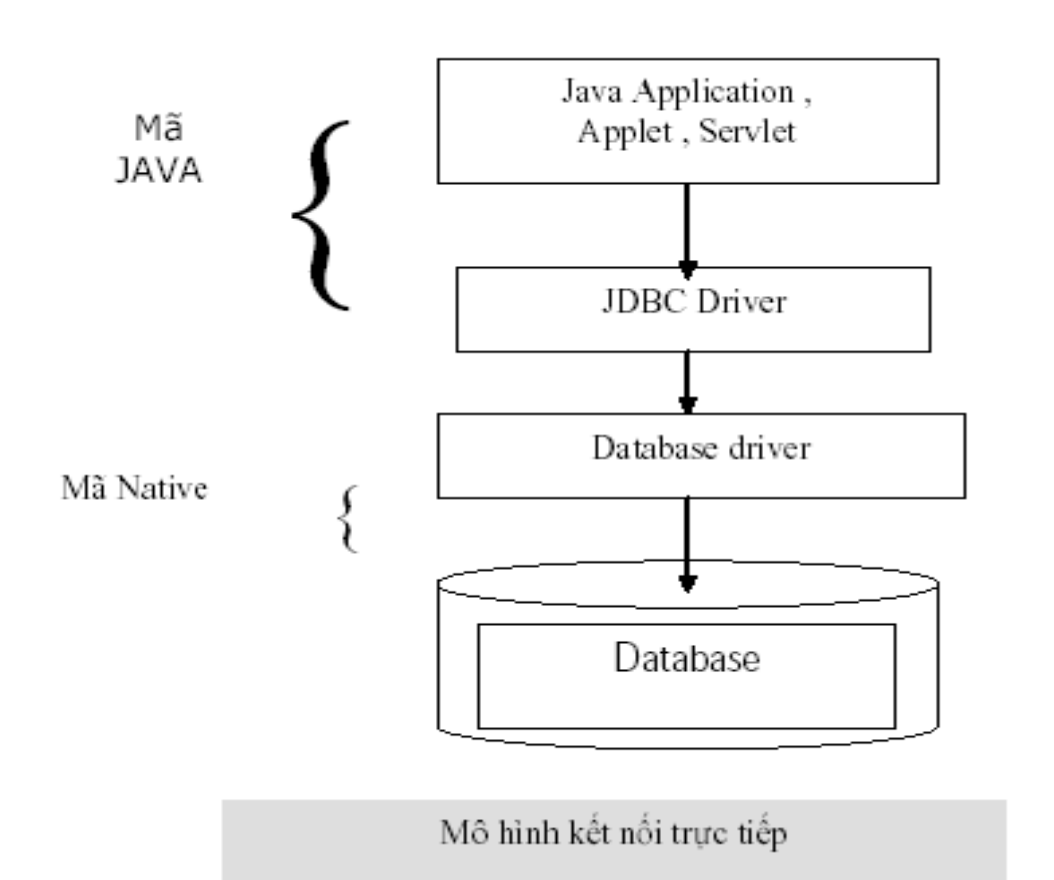
Mô hình truy cập CSDL qua cầu nối JDBC

JDBC KẾT NỐI TRỰC TIẾP VỚI CÁC TRÌNH ĐIỀU KHIỂN CSDL

- Tốt hơn loại 1, loại này cho phép JDBC giao tiếp trực tiếp với các driver hay các hàm API của CSDL.



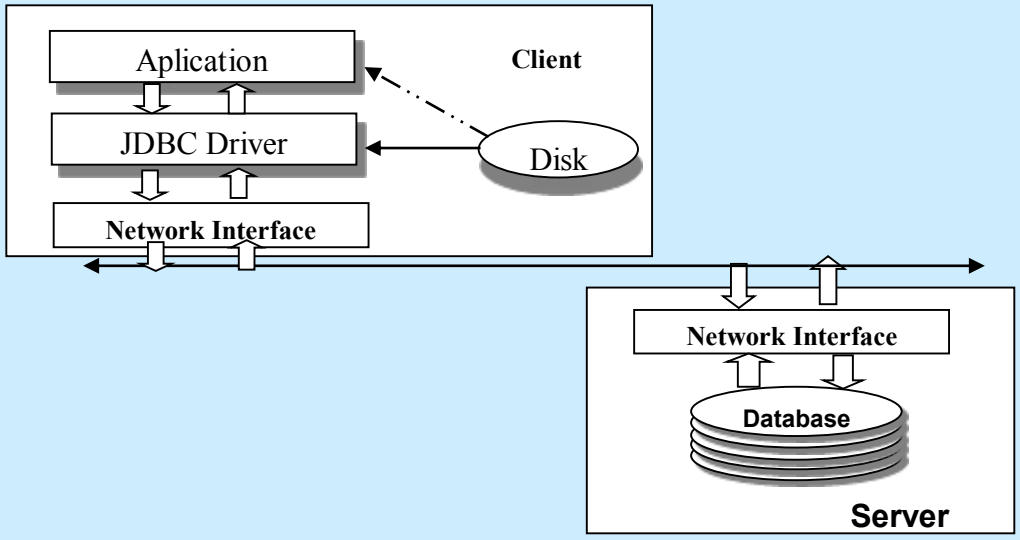
JDBC kết nối trực tiếp với các trình điều khiển cơ sở dữ liệu



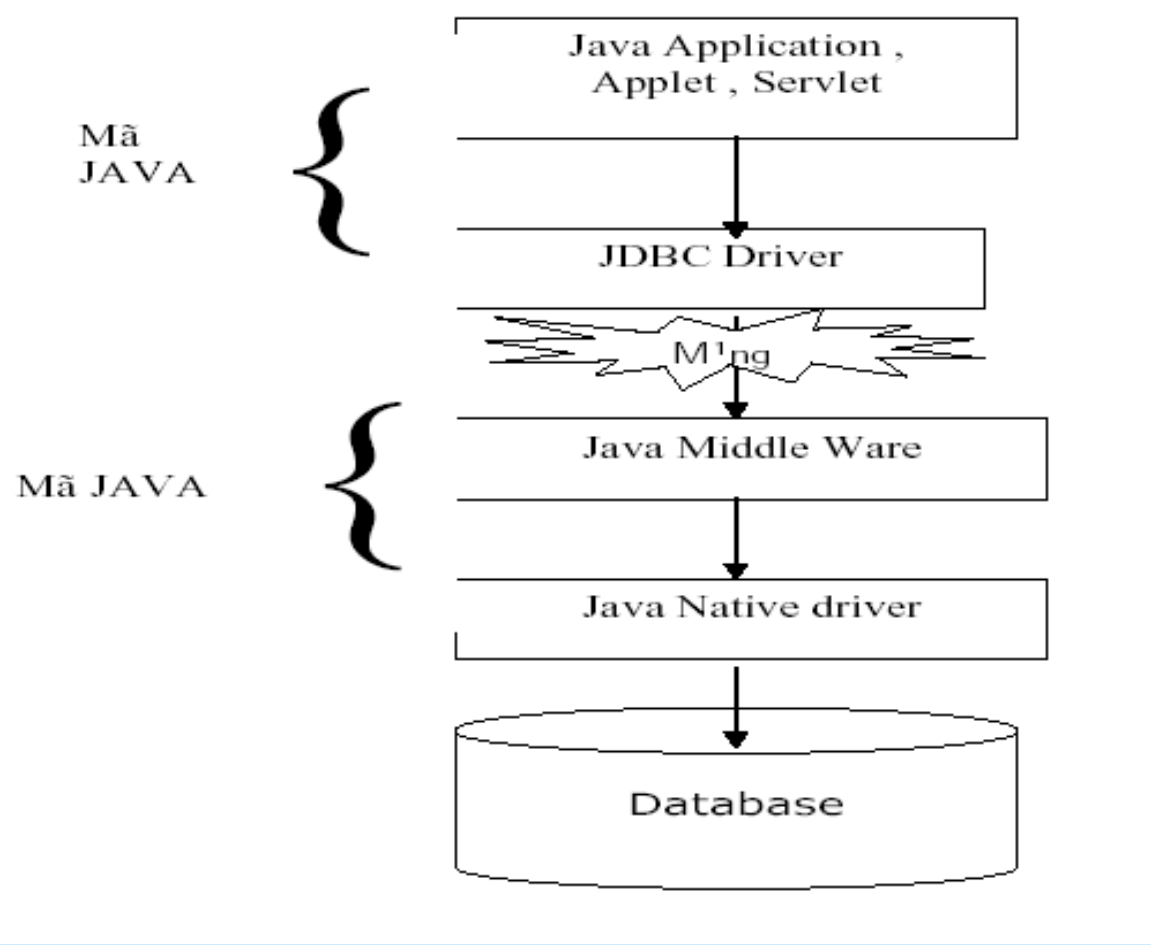
Mô hình kết nối trực tiếp

JDBC KẾT NỐI THÔNG QUA ỨNG DỤNG MẠNG TRUNG GIAN

- 100% java
- Có khả năng giao tiếp trực tiếp với hệ CSDL không cần chuyển đổi

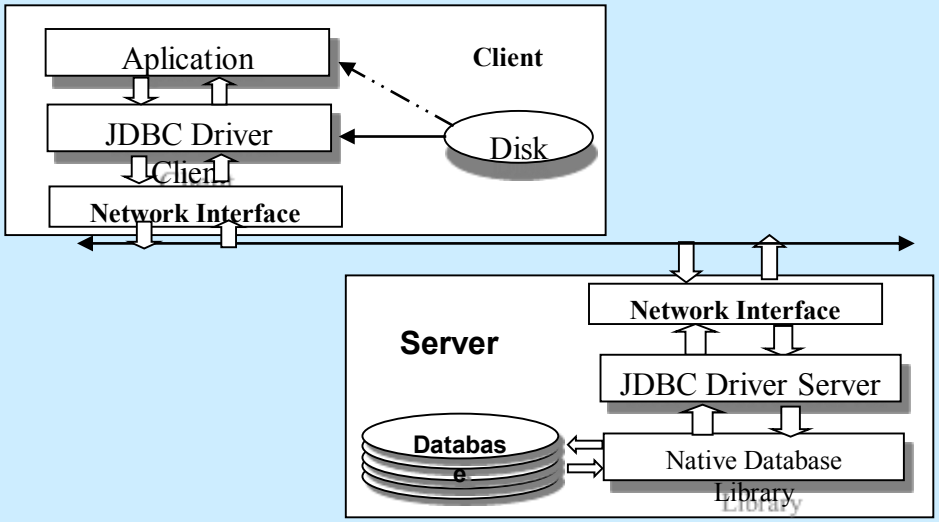


JDBC kết nối thông qua các ứng dụng mạng trung gian

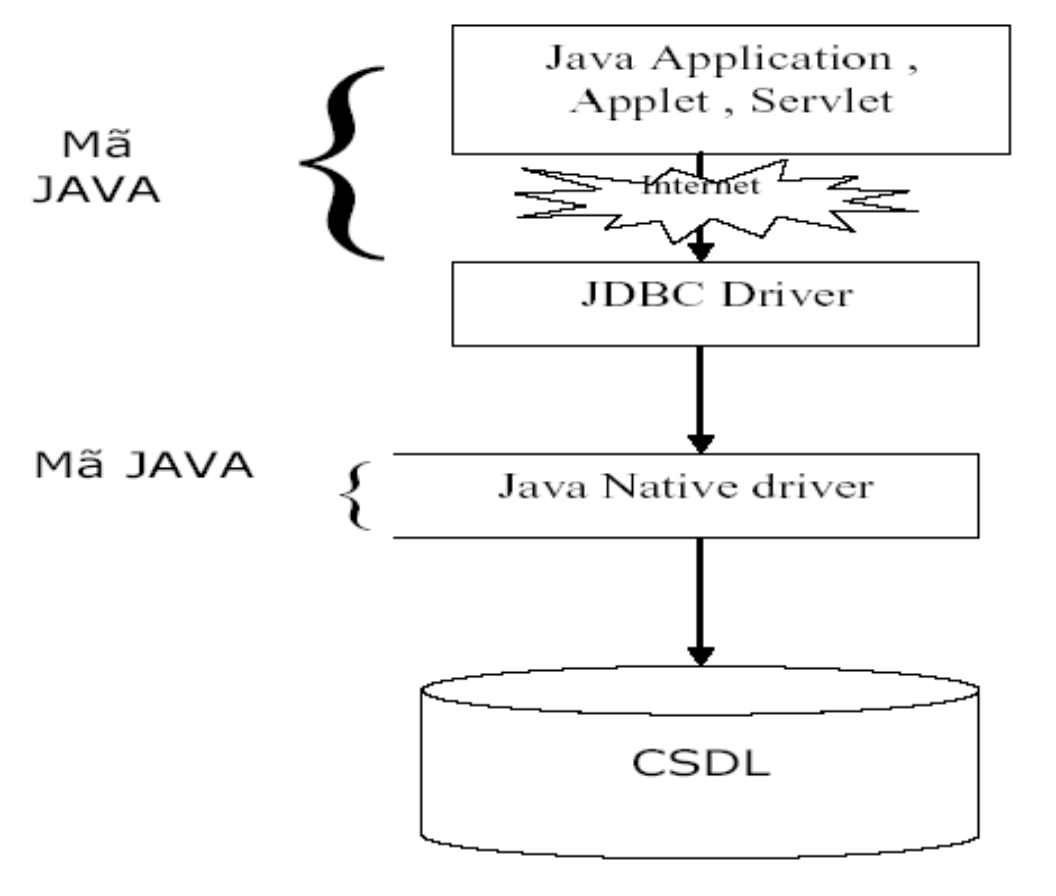


JDBC KẾT NỐI THÔNG QUA CÁC TRÌNH ĐIỀU KHIỂN ĐẶC THÙ Ở XA

- Drivers
  - Có thể chuyển các yêu cầu đến các csdl nằm ở xa.
  - Có thể giao tiếp với nhiều loại CSDL.
  - Không phải của nhà cung cấp csdl
  - Tất cả bằng mã java



JDBC kết nối thông qua các trình điều khiển đặc thù ở xa



Gói Java.sql

- Cung cấp tập hợp các lớp và interface làm việc với CSDL.
- Các lớp
  - DriverManager
  - Date, Time
  - Timestamp
  - Types
- Các Interfaces
  - Driver
  - Connection
  - DatabaseMetaData
  - Statement
  - PreparedStatement
  - CallableStatement
  - ResultSet
  - ResultSetMetaData

Gói Java.sql

Tên giao diện	Mô tả ý nghĩa
CallableStatement	Giao diện chứa các phương thức cho phép ta làm việc với thủ tục lưu trữ nội
DatabaseMetaData	Cho phép ta xem các thông tin về CSDL
PreparedStatement	Giao diện này cho phép ta thực thi các câu lệnh SQL chứa tham số
ResultSetMetaData	
Connection	Thể hiện một kết nối đến CSDL
Driver	Giao diện mà mỗi trình điều khiển phải cài đặt
ResultSet	Thể hiện một tập các bản ghi lấy về từ CSDL
Statement	Giao diện cho phép ta thực hiện các phát biểu SQL

Gói Java.sql

Tên lớp	Ý nghĩa
Date	Lớp biểu diễn kiểu DATE
DriverPropertyInfo	Chứa các thuộc tính của trình điều khiển đã nạp
Timestamp	Lớp biểu diễn cho SQL TimeTemp
DriverManager	Lớp quản lý các trình điều khiên
Time	Lớp biểu diễn kiểu DATE
Types	Lớp định nghĩa các hằng tương ứng với các kiểu dữ liệu SQL, hay còn gọi là kiểu dữ liệu JDBC

Các bước để kết nối CSDL

- Nạp trình điều khiển
- Tạo thông tin kết nối và đối tượng Connection
- Tạo đối tượng Statement để thực thi các lệnh sql..
- Xử lý dữ liệu

Nạp Driver

- Lớp DriverManager chịu trách nhiệm nạp driver và tạo kết nối đến csdl.
- Để nạp và đăng kí trình điều khiển, ta gọi lệnh :

Class.forName(String)

Ví dụ :

DriverManager.registerDriver(new sun.jdbc.odbc.JdbcOdbcDriver());

Hoặc:

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Tương đương

new sun.jdbc.odbc.JdbcOdbcDriver();

Chú ý :Với các trình điều khiển khác nhau thì String của phương thức Class.forName(String) sẽ khác nhau

Tạo thông tin kết nối

- Tiếp tục tạo đối tượng Connection bằng cách gọi phương thức getConnection của lớp DriverManager để yêu cầu trình điều khiển nạp bởi Class.forName() trước đó tiếp nhận thông tin và thực thi kết nối như sau :

```
conn = DriverManager.getConnection(url, “username”, “password ”);
```

Trong đó :

+ url : chuỗi nêu lên đặc điểm csdl có dạng

jdbc:subprotocol:subname

- subprotocol :giao thức con tương ứng với csdl

- subname : tên csdl

+ username : tên đăng nhập csdl

+ password : mật khẩu đăng nhập csdl

Tạo thông tin kết nối

Ví dụ :

Nạp trình điều khiển của MySQL :

```
Class.forName("com.mysql.jdbc.Driver");
Connection conn = DriverManager.getConnection(
"jdbc:mysql://ServrName/DBName?user=UserName&password=Pas
s");
```

Nạp trình điều khiển của SQL :

```
Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");
Connection conn = DriverManager.getConnection("
jdbc:microsoft:sqlserver://ServerName:ServerPort;
DatabaseName=DBName","UserName","Password");
```

Nạp trình điều khiển của Access :

```
Class.forName("sun.jdbc.odbc. JdbcOdbcDriver ");
Connection conn = DriverManager.getConnection("
jdbc:odbc:MyDB");
```

Đối tượng Statement

Tất cả các lệnh tác động đến cơ sở dữ liệu đều có thể thực hiện thông qua một trong 3 đối tượng :

Đối tượng	Mô tả
Satement	Dùng để thực thi các lệnh SQL không có tham số
PreparedStatement	Dùng để thực thi các lệnh SQL có chứa tham số
CallableStatement	Dùng để làm việc với thủ tục lưu trữ nội

Đối tượng Statement

- Đối tượng Connection chứa liên kết trực tiếp đến csdl.
- Sử dụng đối tượng Connection để tạo đối tượng Statement.

  Statement state = con.createStatement();
- Đối tượng này có nhiệm vụ gửi các câu lệnh sql đến csdl.
- **executeQuery(String)** or **executeUpdate(String)** method
- Cùng một đối tượng Statement có thể sử dụng cho nhiều câu lệnh sql khác nhau.

- Có 3 phương thức thực thi
  - executeQuery()
  - executeUpdate()
  - execute()
- The executeQuery()
  - Nhận câu lệnh SQL (select) làm đối số, trả lại đối tượng ResultSet
- ResultSet rs = s.executeQuery(“SELECT * FROM Books”);

- Phương thức executeUpdate()
  - Nhận các câu lệnh sql dạng cập nhật
  - Trả lại số nguyên biểu thị số hàng được cập nhật.
  - UPDATE, INSERT, or DELETE.
- Phương thức execute()
  - Được áp dụng cho trường hợp không rõ loại sql nào được thực hiện.
  - Được áp dụng cho trường hợp câu lệnh sql được tạo ra tự động bởi chương trình.

ResultSet

- Chứa một hoặc nhiều hàng dữ liệu từ việc thực hiện câu lệnh truy vấn.
- Có thể lấy dữ liệu từng hàng dữ liệu một trong ResultSet.
- Sử dụng phương thức next() để di chuyển đến hàng dữ liệu tiếp theo trong ResultSet.
- Hàm next() trả lại true chỉ rằng hàng chứa dữ liệu, trả lại false hàng cuối cùng, không chứa dữ liệu.
- Thực hiện

```
while (rs.next()){
   // examine a row from the results
   }
```

ResultSet

Phương thức	Ý nghĩa
next	Di chuyển con trỏ sang bản ghi kế trong tập bản ghi, phương thức trả về true nếu việc di chuyển là thành công ngược lại cho false
previous	Di chuyển con trỏ về bản ghi trước bản ghi hiện tại, phương thức trả về true nếu việc di chuyển là thành công ngược lại cho false
last	Di chuyển con trỏ về bản ghi cuối cùng trong tập bản ghi, phương thức trả về true nếu việc di chuyển là thành công ngược lại cho false
first	Di chuyển con trỏ về bản ghi đầu tiên trong tập bản ghi, phương thức trả về true nếu việc di chuyển là thành công ngược lại cho false
afterLast	Di chuyển con trỏ về trước bản ghi đầu tiên trong tập b¶n ghi
beforeFirst	Di chuyển con trỏ về sau bản ghi cuối cùng trong tập bản ghi
absolute(int pos)	Di chuyển con trỏ về bản ghi thứ pos tính từ bản ghi đầu tiên nếu pos là số dương, hoặc di chuyển về bản ghi thứ pos tính từ bản ghi cuối cùng nếu pos là số â
relative(int pos)	Di chuyển con trỏ về trước bản ghi hiện tại pos bản ghi nếu pos là số âm, hoặc di chuyển về phía sau pos bản ghi so với bản ghi hiện tại nếu pos là số dương

- Để lấy dữ liệu ở các cột trên mỗi hàng của ResultSet, ta dùng các phương thức
  - gettype(int | String)
    - Đối số là chỉ số cột tính từ 1.
    - Áp dụng cho các cột có kiểu dữ liệu là int, float, Date.....
  - Ví dụ :
    - String isbn = rs.getString(1); // Column 1
    - float price = rs.getDouble(“Price”);

ResultSetMetadata

- Đối tượng này cho biết thông tin về ResultSet

- ***ResultSet rs = stmt.executeQuery(SQLString);***
  ResultSetMetaData rsmd = rs.getMetaData();
  int numberOfColumns = rsmd.getColumnCount();
- ***getColumnName(int column)***

Database Metadata

- Đối tượng này cho biết thông tin về csdl.

Chương trình mẫu

```
import java.sql.*;
import javax.sql.*;
public class Connect {
public static void main(String args[]) throws
ClassNotFoundException,SQLException {
System.out.println("Ket noi CSDL");
try{
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
String url="jdbc:odbc:myDatabase";
Connection conn=DriverManager.getConnection(url,"login","password");
Statement stmt=conn.createStatement();
String sql0="INSERT INTO KhachHang(Id,TenKH,DiaChi,Luong)
VALUES('8',‘Nguyen C',‘HCM','900')";
stmt.executeUpdate(sql0);
//Cap nhat du lieu
String sql2="UPDATE Khachhang SET Luong=Luong+luong*0.2";
int n=stmt.executeUpdate(sql2);
if (n < 1) System.out.println("Khong co ban ghi nao duong cap nhat");
else System.out.println("Co "+ n +" ban ghi duong cap nhat");
```

Chương trình mẫu

```
String sql="SELECT Id,TenKH,DiaChi,Luong FROM KhachHang";
ResultSet rs=stmt.executeQuery(sql);
while (rs.next())
{ int id=rs.getInt("Id");
double l=rs.getDouble("Luong");
String s=rs.getString("TenKH");
String d=rs.getString("DiaChi");
System.out.println("ID=" +id +" " + s+ " " + d + " Luong=" + l) ;
}
} catch(SQLException e) {System.out.println("Loi thao tac CSDL");}
}
}
```

Chương trình mẫu

```
import java.sql.*;
class JDBCDemo1 {
     public static void main(String[] args) {
  try{
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
       Connection con=DriverManager.getConnection("jdbc:odbc:Accserver");
      Statement stmt = con.createStatement();
     String sql="Select * from Table1";
     ResultSet rs = stmt.executeQuery(sql);
     ResultSetMetaData rsmd = rs.getMetaData();
     int numberOfColumns = rsmd.getColumnCount();
     for(int j=1; j<=numberOfColumns;j++) {
              System.out.println(rsmd.getColumnLabel(j)); }
     while(rs.next()) {
              for(int i=1; i<=numberOfColumns;i++){
              System.out.println(rs.getObject(i));   }}
              rs.close();
              stmt.close();
     } catch(Exception e){  System.out.println("Error " + e); }
       }
}
```