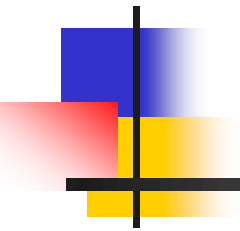
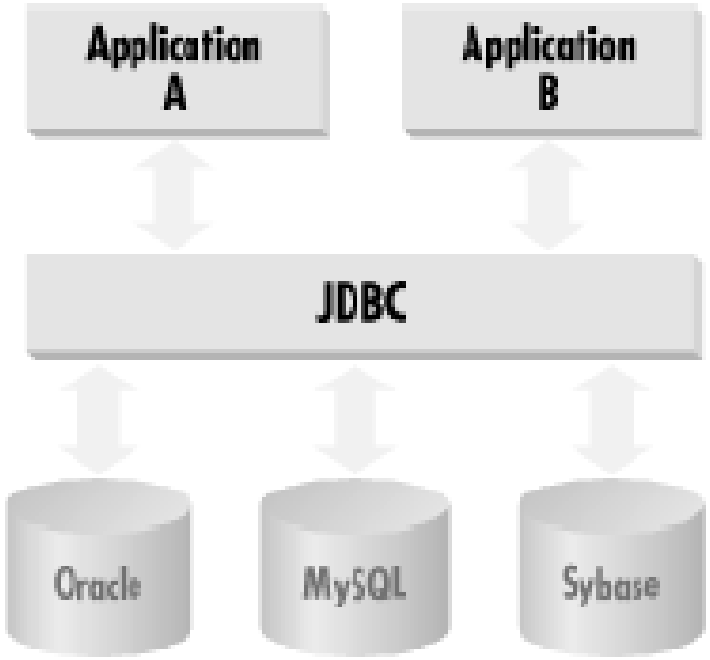


Bài 9

Lập Trình Cơ Sở Dữ Liệu

Tổng Quan

- JDBC cung cấp tập các lớp và interface cho phép chương trình Java có thể làm việc được với hệ cơ sở dữ liệu
- Tập các lớp của JDBC có thể làm việc được với mọi hệ quản trị cơ sở dữ liệu



Database Driver

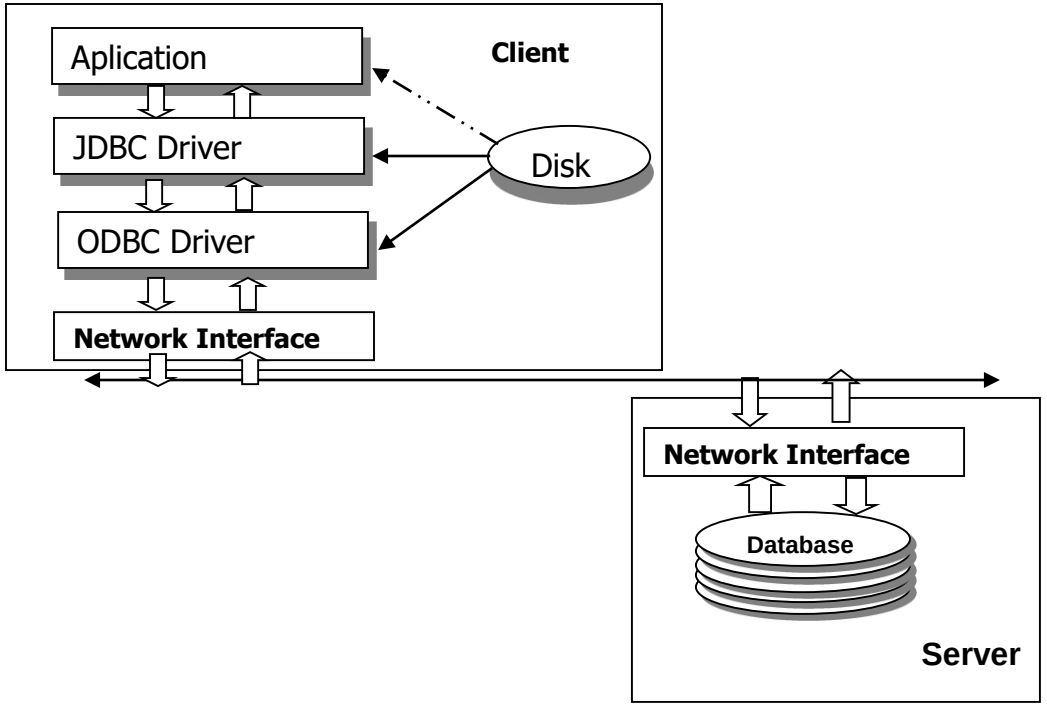
- Bảo đảm ứng dụng java tương tác với mọi cơ sở dữ liệu dưới một cách thức chuẩn và duy nhất.
- Bảo đảm những yêu cầu từ chương trình sẽ được biểu diễn trong cơ sở dữ liệu dưới một ngôn ngữ mà cơ sở dữ liệu hiểu được
- Nhận các yêu cầu từ client, chuyển nó vào định dạng mà cơ sở dữ liệu có thể hiểu được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

JDBC Driver

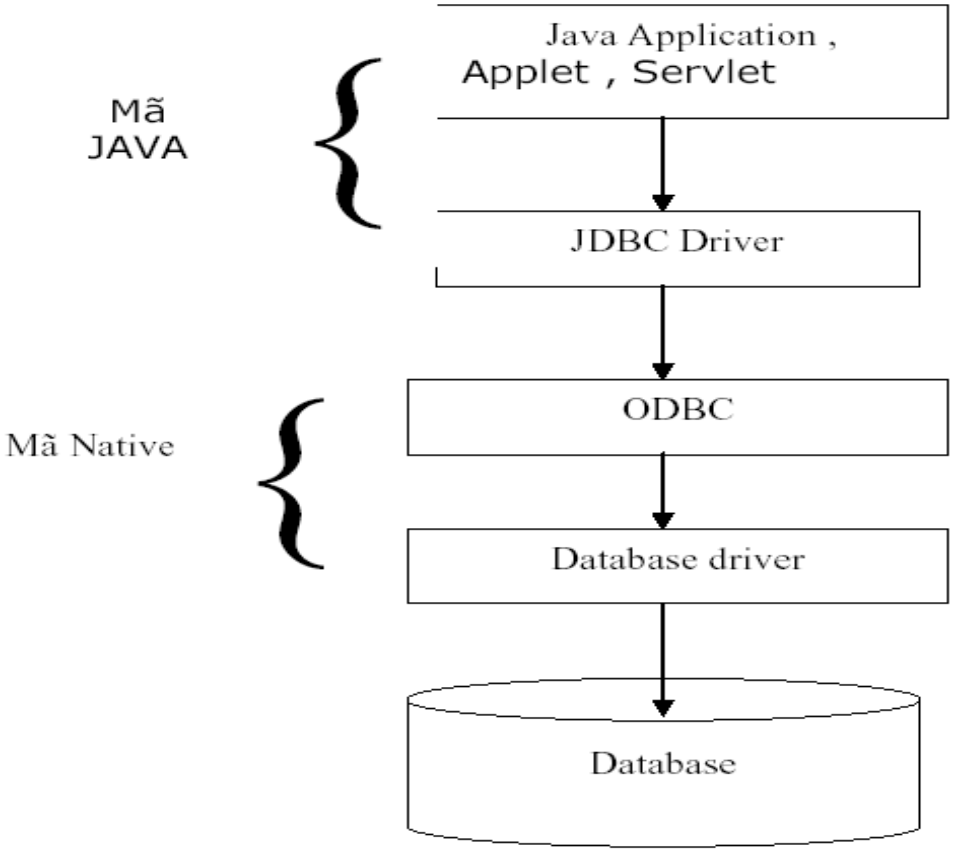
- Có 4 loại JDBC Driver
  - Loại 1 : JDBC sử dụng cầu nối ODBC
  - Loại 2 : JDBC kết nối trực tiếp với các trình điều khiển
  - Loại 3 : JDBC kết nối thông qua các ứng dụng mạng trung gian
  - Loại 4 : JDBC kết nối thông qua các trình điều khiển đặc thù ở xa
- Loại 2,3,4 nói chung được viết bởi nhà cung cấp cơ sở dữ liệu, 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.



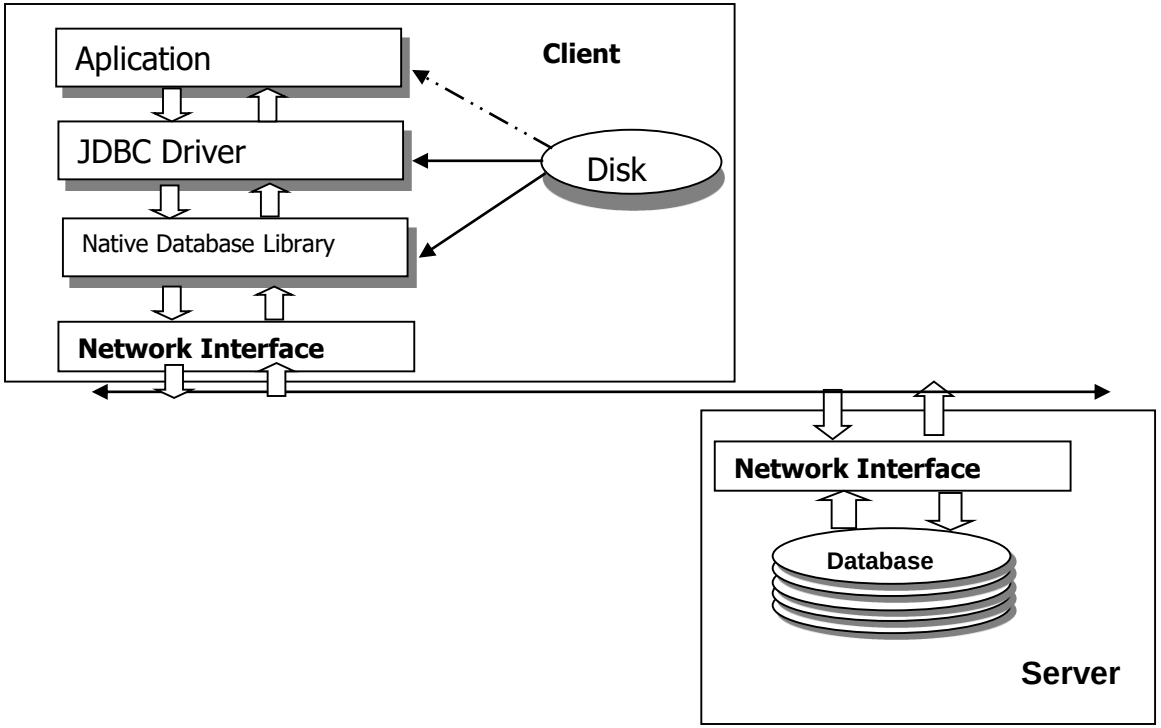
JDBC sử dụng cầu nối ODBC



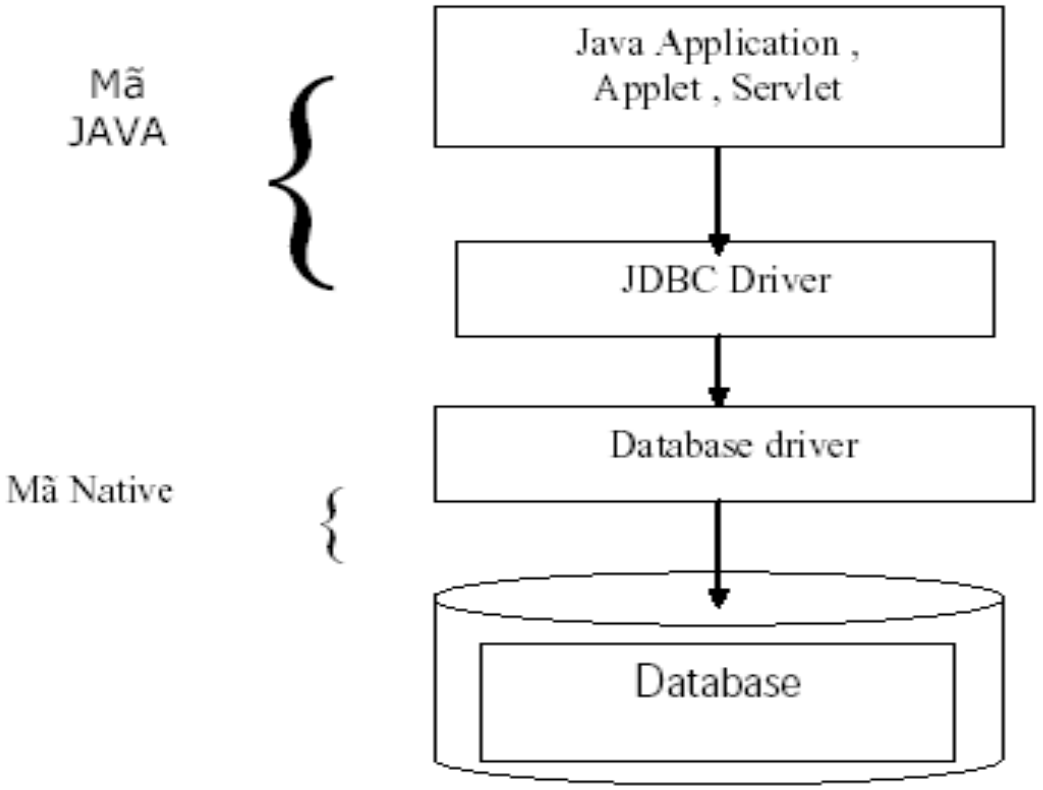
Mô hình truy cập CSDL qua cầu nối JDBC

JDBC KẾT NỐI TRỰC TIẾP VỚI CÁC TRÌNH ĐIỀU KHIỂN CƠ SỞ DỮ LIỆU

- Loại này cho phép JDBC giao tiếp trực tiếp với các driver hay các hàm API của cơ sở dữ liệu.



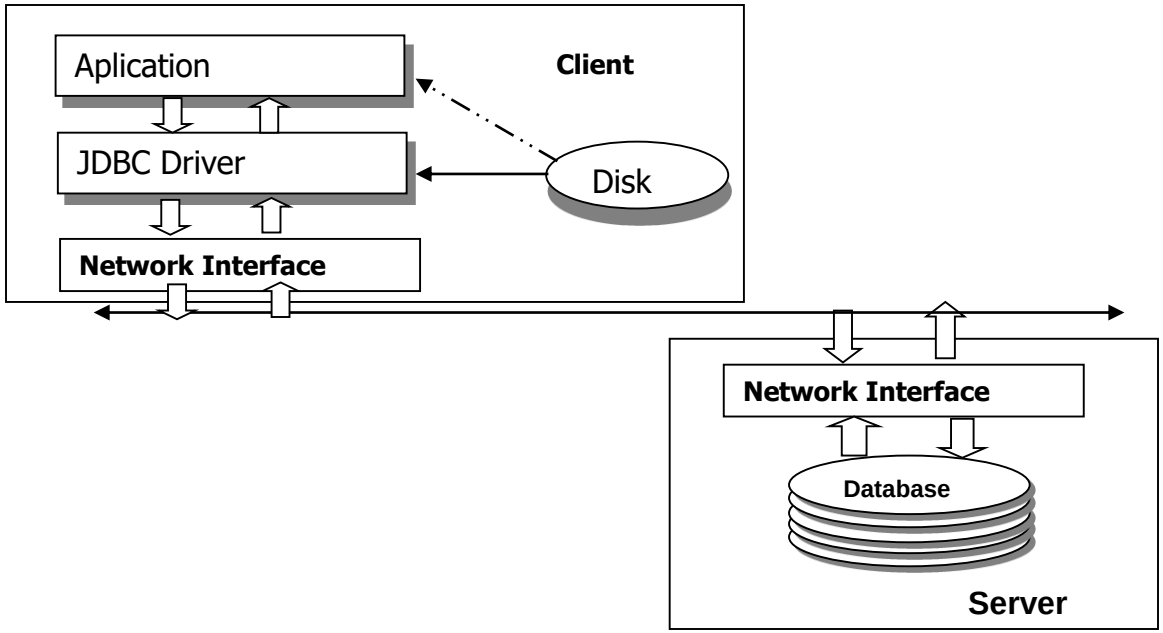
JDBC kết nối trực tiếp với các trình điều khiển cơ sở dữ liệu



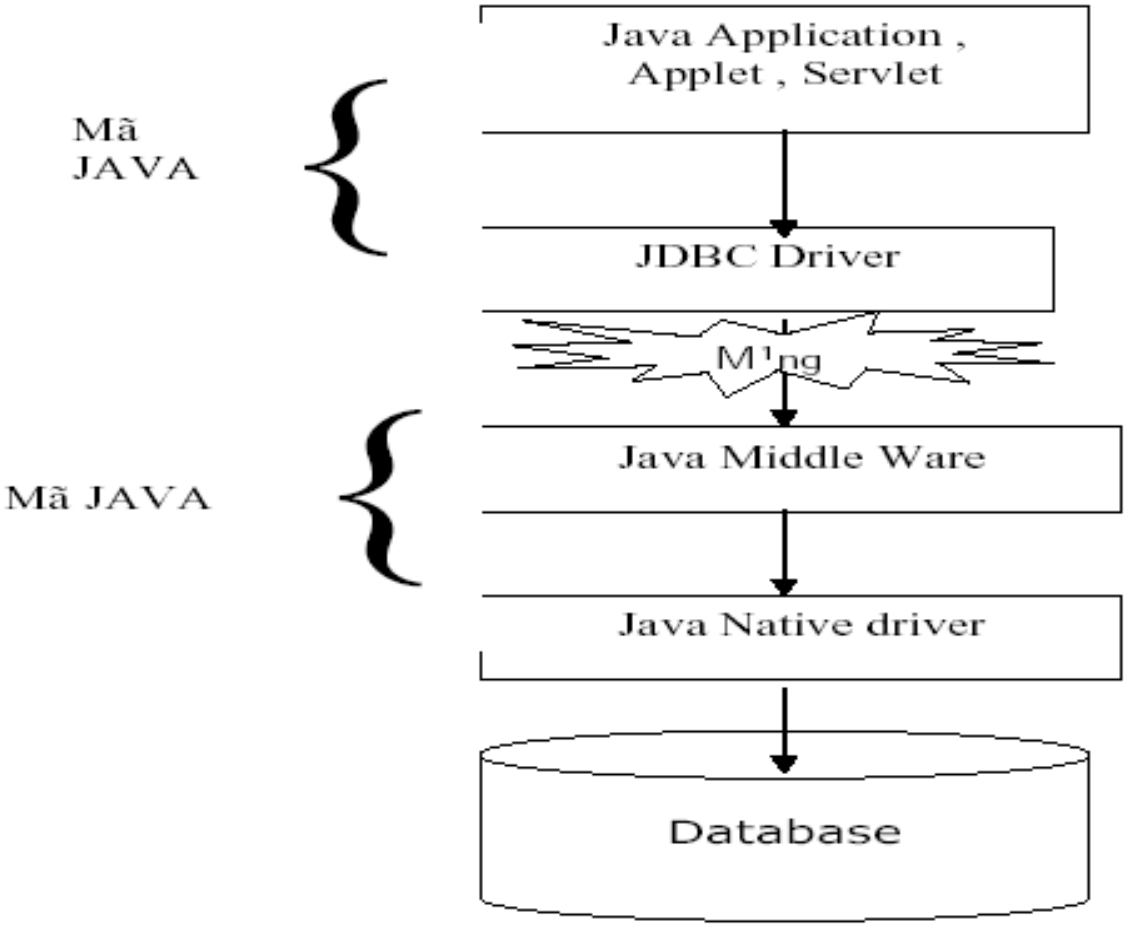
Mô hình kết nối trực tiếp

JDBC KẾT NỐI THÔNG QUA ỨNG DỤNG MẠNG TRUNG GIAN

- 100% java
- Có khả năng giao tiếp trực tiếp với hệ cơ sở dữ liệu không cần chuyển đổi

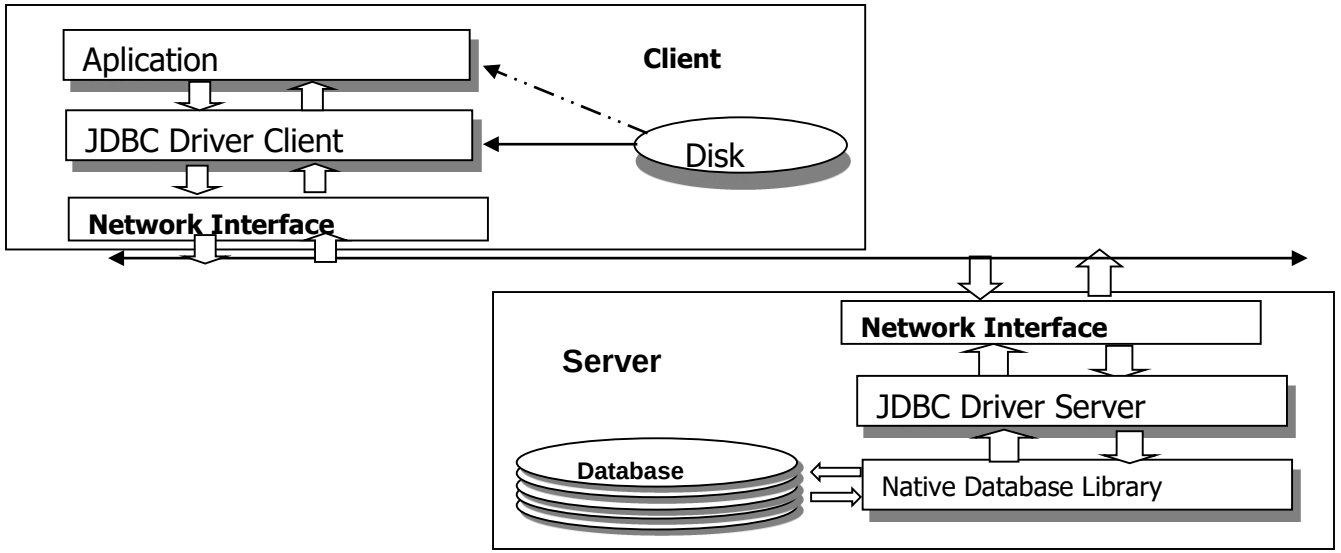


JDBC KẾT NỐI THÔNG QUA ỨNG DỤNG MẠNG TRUNG GIAN

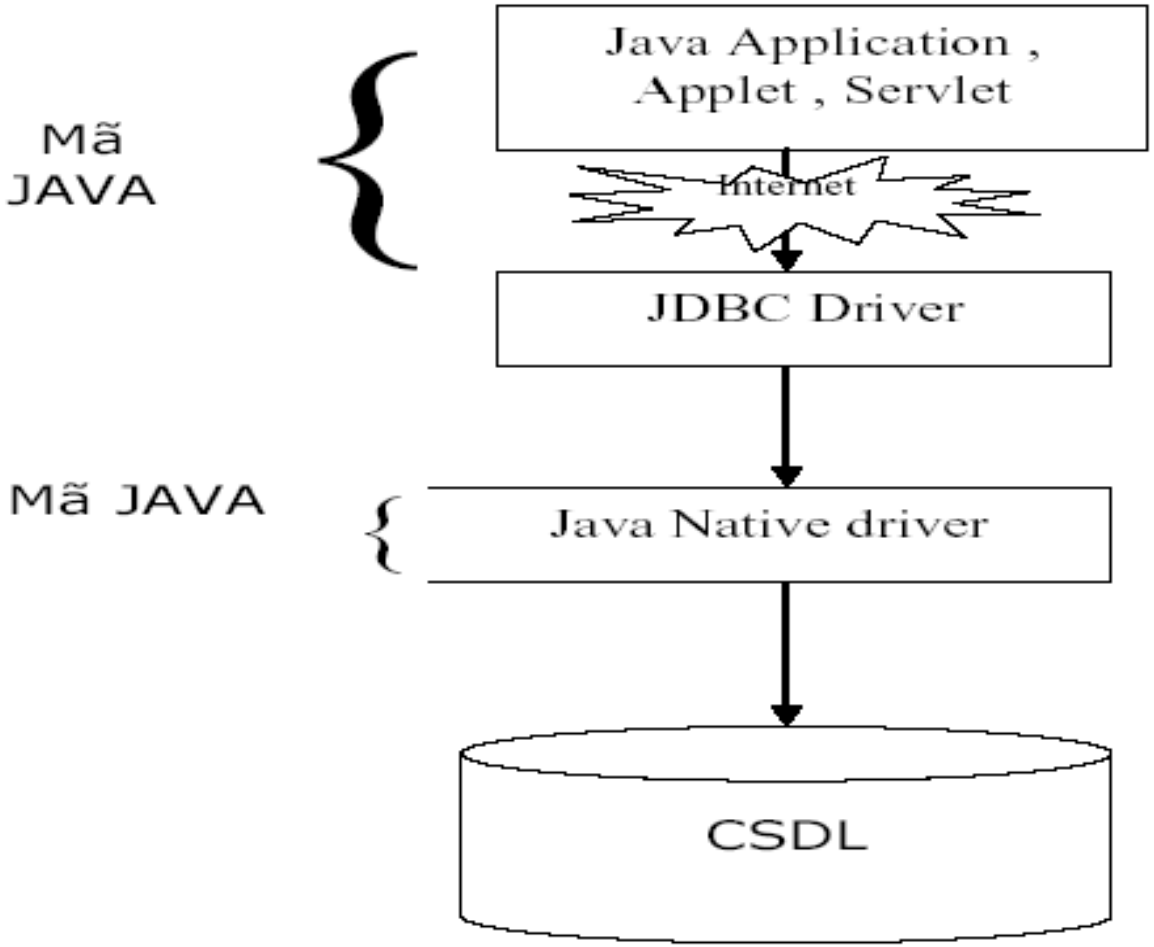


JDBC KẾT NỐI THÔNG QUA CÁC TRÌNH ĐIỀU KHIỂN ĐẶC THÙ Ở XA

- 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

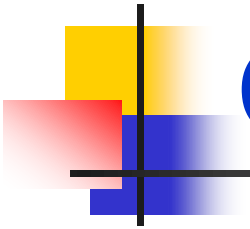


JDBC KẾT NỐI THÔNG QUA CÁC TRÌNH ĐIỀU KHIỂN ĐẶC THÙ Ở XA



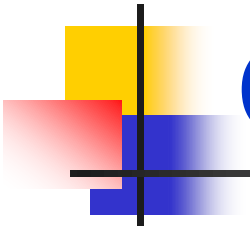
Gói Java.sql

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



Gói Java.sql

- CallableStatement : Giao diện chứa các phương thức cho phép làm việc với các thủ tục lưu trữ
- DatabaseMetaData : Cho phép xem các thông tin về cơ sở dữ liệu
- PreparedStatement : Giao diện cho phép thực thi các câu lệnh SQL chứa tham số



Gói Java.sql

- Connection : thể hiện một kết nối đến cơ sở dữ liệu
- Driver: giao diện mà các trình điều khiển phải cài đặt
- ResultSet : thể hiện một tập các bản ghi lấy về từ cơ sở dữ liệu
- Statement : giao diện cho phép ta thực thi các phát biểu SQL

Gói Java.sql

- 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 khiển
- Time : lớp biểu diễn kiểu Time
- 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 cơ sở dữ liệu

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

Các bước để kết nối cơ sở dữ liệu

Step	Description	Use (java.sql package)	Methods
1	**Load JDBC Driver**	Java.lang.Class	forName(…)
2	**Establish a DB connection**	java.sql.Connection java.sql.DriverManager	DriverManager getConnection(…) → Connection
3	**Create & execute SQL statements**	java.sql.Statement java.sql.PrepareStatement java.sql.CallableStatement	execute(…) executeQuery(…) → SELECT executeUpdate(…) → INSERT/UPDATE/DELETE
4	**Process the results**	java.sql.ResultSet	first(), last(), next(), previous() getXXX(..)
5	**Close**	ResultSet, Statement,	close()

Nạp Driver

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

Class.forName(String)

Ví dụ **:**

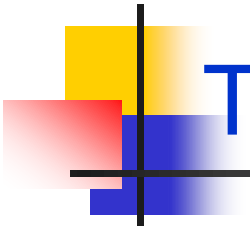
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Hoặc:

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

Hoặc:

new sun.jdbc.odbc.JdbcOdbcDriver();



Tạo thông tin kết nối

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

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

Tạo thông tin kết nối

Trong đó :

+ url : chuỗi nêu lên đặc điểm csdl có dạng

jdbc:subprotocol:subname

- subprotocol :giao thức con tương ứng với cơ sở dữ liệu

- subname : tên cơ sở dữ liệu

+ username : tên đăng nhập

+ password : mật khẩu đăng nhập

Tạo thông tin kết nối

Nạp trình điều khiển của Access :

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

Connection conn = DriverManager.getConnection(" jdbc:odbc:DBName“,”username”,”password”);

Nạp trình điều khiển của MySQL :

Class.forName("com.mysql.jdbc.Driver");

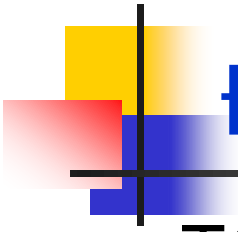
Connection conn = DriverManager.getConnection(

"jdbc:mysql://ServrName:3306/DBName”,”userName”,”password");

Nạp trình điều khiển của SQL Server :

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

Connection conn = DriverManager.getConnection(" jdbc:sqlserver://ServerName:ServerPort; DatabaseName=DBName",“userName",“password");



Đối tượng Statement

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

- Statement : Thực thi các câu lệnh SQL không có tham số
- PreparedStatement : Thực thi các câu lệnh SQL có chứa tham số
- CallableStatement : Làm việc với các thủ tục lưu trữ

Đối tượng Statement

- Đối tượng Connection chứa liên kết trực tiếp đến cơ sở dữ liệu.
- Sử dụng đối tượng Connection để tạo đối tượng Statement.

```
Statement stmt = conn.createStatement();
```

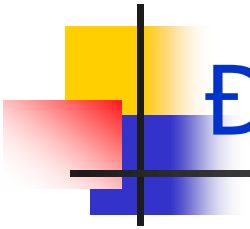
- Đối tượng này có nhiệm vụ gửi các câu lệnh sql đến cơ sở dữ liệu
- Cùng một đối tượng Statement có thể sử dụng cho nhiều câu lệnh sql khác nhau.

Đối tượng Statement

- Có 3 phương thức thực thi :
  - executeQuery()
  - executeUpdate()
  - execute()
- Phương thức executeQuery()
  - Nhận câu lệnh SQL (select) làm đối số, trả lại đối tượng ResultSet

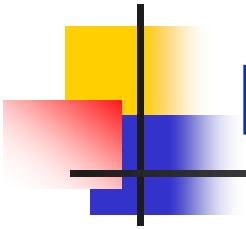
Ví dụ :

```
ResultSet rs = stmt.executeQuery("SELECT * FROM Books");
```



Đối tượng Statement

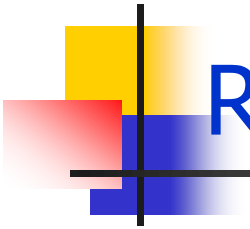
- Phương thức executeUpdate()
  - Nhận các câu lệnh sql dạng cập nhật
  - Trả lại số nguyên biểu thị số hàng được cập nhật.
  - UPDATE, INSERT, DELETE
- Phương thức execute()
  - Được áp dụng cho trường hợp không rõ loại sql nào được thực hiện.
  - Được áp dụng cho trường hợp câu lệnh sql được tạo ra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



ResultSet

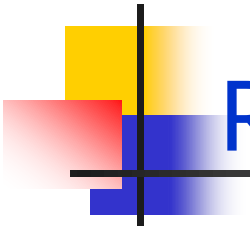
- Chứa một hoặc nhiều hàng dữ liệu từ việc thực hiện câu lệnh truy vấn.
- Có thể lấy dữ liệu từng hàng dữ liệu một trong ResultSet.
- Sử dụng phương thức next() để di chuyển đến hàng dữ liệu tiếp theo trong ResultSet.
- Hàm next() trả lại true chỉ rằng hàng chứa dữ liệu, trả lại false hàng cuối cùng,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

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



ResultSet

- next : di chuyển con trỏ sang tập bản ghi kế tiếp, trả về true nếu thành công, ngược lại false
- previous : di chuyển con trỏ về bản ghi trước bản ghi hiện tại
- last : di chuyển con trỏ về bản ghi cuối cùng trong tập bản ghi
- first : di chuyển con trỏ về bản ghi đầu tiên trong tập bản ghi



ResultSet

- beforeFirst : di chuyển con trỏ về trước bản ghi đầu tiên trong tập bản ghi
- afterLast : 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 tính từ bản ghi cuối cùng nếu pos là số âm)
- 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

ResultSet

- 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");// column 2

Chương trình mẫu

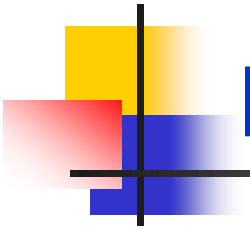
```
import java.sql.*;
public class Connect {
public static void main(String args[]) throws
ClassNotFoundException,SQLException {
System.out.println("Ket noi CSDL");
try{
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
String url="jdbc:odbc:DBName";
Connection
conn=DriverManager.getConnection(url,"user","password");
Statement stmt=conn.createStatement();
```

Chương trình mẫu

```
String sql1="INSERT INTO TableName(Id,TenKH,DiaChi,Luong) VALUES('8',`Nguyen C',`HCM','900')";
stmt.executeUpdate(sql1);
//Cap nhat du lieu
String sql2="UPDATE TableName SET Luong=Luong+Luong*0.1";
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 duoc cap nhat");
```

Chương trình mẫu

```
String sql="SELECT  Id,TenKH,DiaChi,Luong FROM TableName";
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(“Error");}
}
}
```

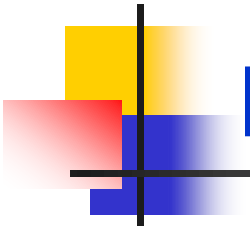


ResultSetMetadata

- 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

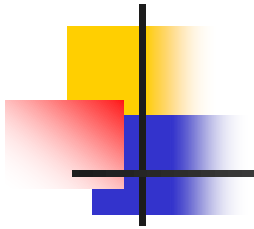
- Đối tượng này cho biết thông tin về cơ sở dữ liệu.

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:Database");
    Statement stmt = con.createStatement();
   String sql="Select * from Table_name";
   ResultSet rs = stmt.executeQuery(sql);
   ResultSetMetaData rsmd = rs.getMetaData();
   int numberOfColumns = rsmd.getColumnCount();
```

Chương trình mẫu

```
    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);
    }
 }
}
```



Hết *!!!*