LẬP TRÌNH JAVA

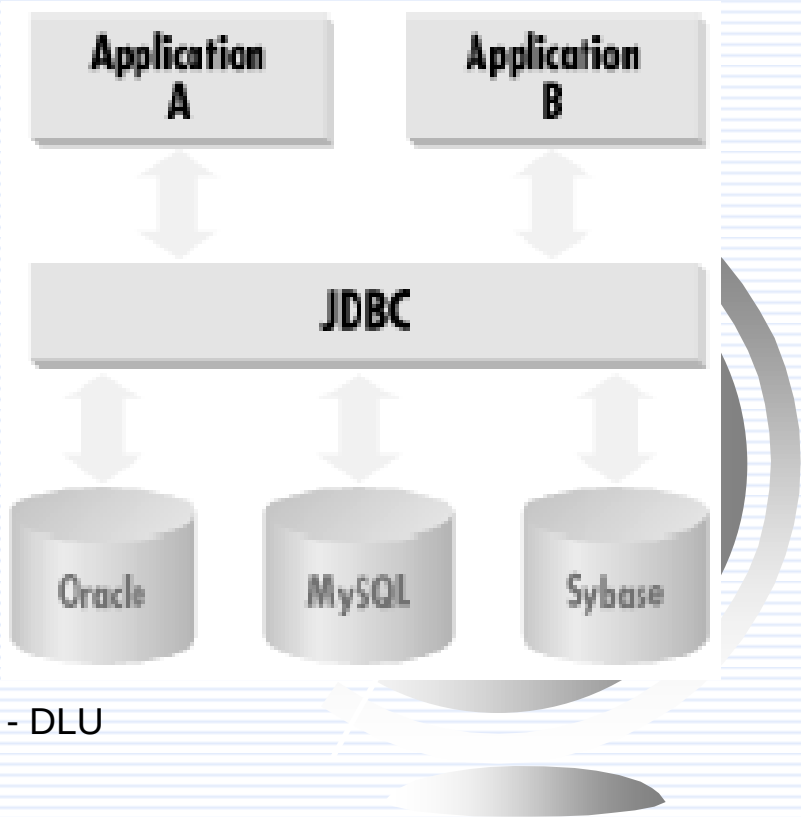
CHƯƠNG 7: LẬP TRÌNH CƠ SỞ DỮ LIỆU

Nội dung

- JDBC là gì?
- ODBC là gi?
- Kết nối CSDL (SQL Server, MySQL, Derby, …)
- Truy vấn và cập nhật dữ liệu

JDBC

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



Ví dụ kết nối JDBC

```
try {
    1. Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    2. Connection con =
       DriverManager.getConnection("jdbc:odbc:Northwind");
    3. Statement stmt = con.createStatement();
       :
       :
       :
       :
}
catche(...)
{
}
```

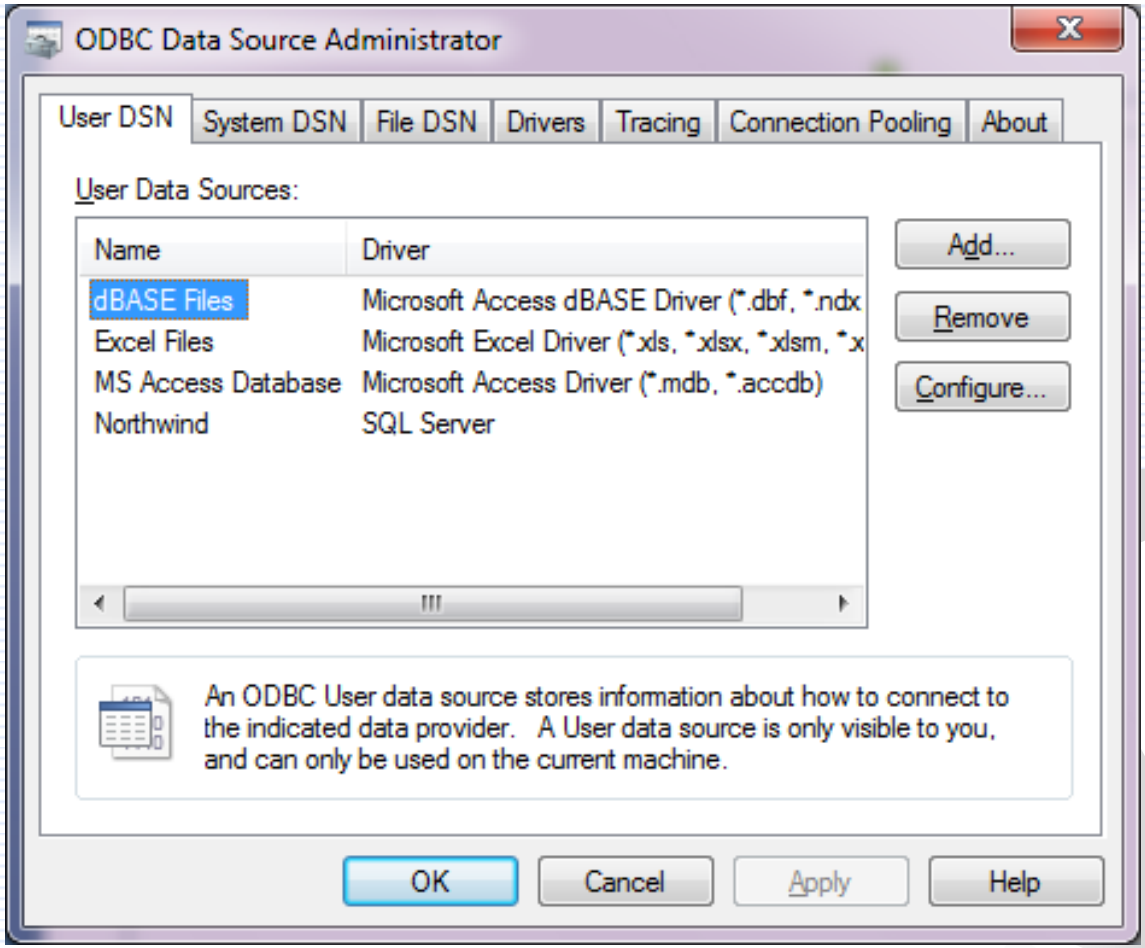
Database URL

- Database URL:
  - Là một chuỗi được dùng để kết nối csdl.
- Cú pháp :

  jdbc:subprotocol name:other_stuff
  - **subprotocol name**: được dùng tuỳ vào loại driver sử dụng để kết nối csdl.
    - Ví dụ : subprotocol name là **odbc** nếu driver là cầu nối jdbcodbc
  - **Other_stuff:** cũng phụ thuộc vào loại driver nào được sử dụng. ví dụ nếu driver là cầu nối jdbcodbc thì thành phần này là tên của đối tượng ODBC

ODBC

- ✓ Lệnh: Run\odbcad32.exe



Kết nối CSDL

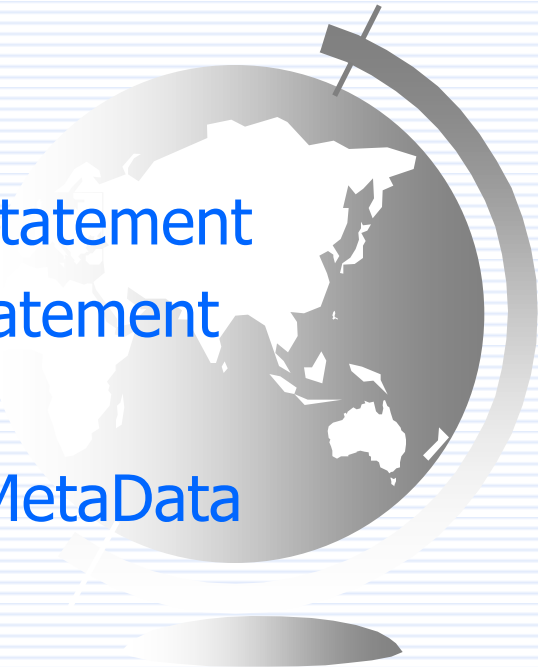
- Có 3 bước chính để kết nối CSDL trong Java:
  - Nạp database drivers
  - Tạo nối kết, Tạo đối tượng Connection
  - Tạo đối tượng Statement để thực thi các lệnh sql..

Nạp Driver và kết nối CSDL

- Lớp DriverManager chịu trách nhiệm nạp driver và tạo kết nối đến csdl:
  - ***DriverManager.registerDriver(new sun.jdbc.odbc.JdbcOdbcDriver());***
- Hoặc dùng lớp Class:
  - ***Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");***
- Sử dụng lớp DriverManager:
  - ***Connection cnn = DriverManager.getConnection("jdbc:odbc:Northwind", "sa", "sa");***

Gói Java.sql

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



Đối tượng Statement

- Sử dụng đối tượng Connection để tạo đối tượng Statement.
  - **Statement s = con.createStatement();**
- Đối tượng này có nhiệm vụ gửi các câu lệnh sql đến csdl.
  - **executeQuery(String)** hoặc **executeUpdate(String)**

Đối tượng Statement

- Có 3 phương thức thực thi
  - executeQuery()
  - executeUpdate()
  - execute()
- executeQuery()
  - Nhận câu lệnh SQL (select) làm đối số, trả lại đối tượng ResultSet
  - Ví dụ: ResultSet rs = s.executeQuery("SELECT * FROM CUSTOMERS");

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, 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 đwocj tạo ra tự động bởi chương trình.

ResultSet

- Chứa một hoặc nhiều hàng dữ liệu từ việc thực hiện câu lệnh truy vấn.
- 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 không chứa dữ liệu.
- Ví dụ:

```
while (rs.next()){
    ...
}
```

ResultSet

- Để lấy dữ liệu ở các cột trên mỗi hàng của ResultSet, ta dùng các phương thức.
  - get**Type**(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");

ResultSet Metadata

- 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)***

Các bước thực hiện JDBC

1. Importing Packages
2. Registering the JDBC Drivers
3. Opening a Connection to a Database
4. Creating a Statement Object
5. Executing a Query and Returning a Result Set Object
6. Processing the Result Set
7. Closing the Result Set and Statement Objects
8. Closing the Connection

1. Importing Packages

- **import** java.sql.Connection;
- **import** java.sql.DriverManager;
- **import** java.sql.ResultSet;
- **import** java.sql.ResultSetMetaData;
- **import** java.sql.Statement;

2. Registering JDBC Drivers

- Class.*forName*(*DBDriver*);
  - *DBDriver* =
    "sun.jdbc.odbc.JdbcOdbcDriver";

3: Opening connection to a Database

- DriverManager.*getConnection*(*DBSource*,*sqlUser*,*sqlPass*);
  - *DBSource* = "jdbc:odbc:NorthWind";
  - *sqlUser* = "sa";
  - *sqlPass* = "sa";
- ✓ Return a *Connection* object.
  - Connection *myConnection;*
  - *myConnection* = DriverManager.*getConnection*(*DBSource*,*sqlUser*,*sqlPass*);

4. Creating a Statement Object

- ✓ Call createStatement() method:
  - *myConnection*.createStatement();
- ✓ Return a Statement object:
  - Statement stmt;
  - *stmt = myConnection*.createStatement();

5. Executing a Query, Returning a Result Set Object
6. Processing the Result Set

- ResultSet rs;
- ResultSetMetaData rsmd;
- *rs* = *stmt*.executeQuery(strSQL);
- *rsmd* = *rs*.getMetaData();
- **int** col = *rsmd*.getColumnCount();
- **while**(*rs*.next()){
  - **for**(**int** i=1; i<=col; i++){
  - System.*out*.print(*rs*.getString(i)+ "\t");
- }
- System.*out*.println();

7. Closing the Result Set and Statement Objects
8. Closing the Connection

- ✓rs.close();
- ✓stmt.close();
- ✓myConnection.close();

CSDL Derby, MySQL

- Derby:
  - Driver:

Class.forName("org.apache.derby.jdbc.EmbeddedDriver");

  - Connection:

DriverManager.getConnection("jdbc:derby://localhost:1527/databasename", "user", "pass");

- MySQL:
  - Driver:

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

  - Connection:

DriverManager.getConnection("jdbc:mysql://localhost:3306/databasename","user","pass"");

```
1.  public class DatabaseConnection {
2.      Connection conn = null;
3.      String url = "jdbc:mysql://localhost:3306/";
4.      String dbName = "testctk37";
5.      String driver = "com.mysql.jdbc.Driver";
6.      String userName = "root";          String password = "";
7.      Statement stmt;
8.      ResultSet rs;
9.      ResultSetMetaData rsmd;
10.     public DatabaseConnection() {
11.         try {
12.             Class.forName(driver).newInstance();
13.             conn = DriverManager.getConnection(url + dbName, userName, password);
14.             stmt = conn.createStatement();
15.         } catch (SQLException e) {
16.             System.err.println(e.getMessage());
17.         }
18.     }
19. }
```

Update Query

- With a strSQL update string:
  - *Example strSQL* = "Insert Into Table Values(val1, val2, …)";
  - *stmt*.executeUpdate(strSQL);
- Use StoreProcedure:
  - Replace strSQL by StoreProcedure

Store Procedure

- Use CallableStatement
  - CallableStatement cs;
- Call a procedure with no parameters
  - cs = *connection*.prepareCall(*"{call myproc}"*);
  - cs.execute();
- Call a procedure with one IN parameter
  - cs = *connection*.prepareCall(*"{call myprocin(?)}"*);
  - cs.setString(1, *"a string"*);
  - cs.execute();

Store Procedure

- ✓ Call a procedure with one OUT parameter
  - cs = *connection*.prepareCall(*"{call myprocout(?)}"*);
- ✓ Register the type of the OUT parameter
  - cs.registerOutParameter(1, *Types.VARCHAR*);
  - cs.execute();
  - String outParam = cs.getString(1); // OUT parameter