



**UIT**  
TRƯỜNG ĐẠI HỌC  
CÔNG NGHỆ THÔNG TIN

# **NGÔN NGỮ LẬP TRÌNH JAVA**

## **CHƯƠNG 7: LẬP TRÌNH CSDL VỚI JDBC (P2)**

GVGD: ThS. Lê Thanh Trọng

# Nội dung

- ❖ Mục tiêu bài học
- ❖ Statement
- ❖ ResultSet
- ❖ PreparedStatement
- ❖ ResultSetMetaData
- ❖ DatabaseMetaData
- ❖ Close the connection
- ❖ Tóm tắt bài học

# Mục tiêu bài học

- ❖ Giới thiệu cho sinh viên các bước thao tác với CSDL trong Java sau khi đã kết nối thành công ở phần trước.
- ❖ Sử dụng Statement và ResultSet để kết nối và thao tác cơ bản hoặc sử dụng PreparedStatement cho các truy vấn có tham số.
- ❖ Ngoài ra là việc lấy thông tin về kết quả truy vấn hoặc database bằng ResultSetMetaData và DatabaseMetaData.
- ❖ Cách đóng kết nối đến CSDL.

# Các bước kết nối CSDL

- ❖ Có 5 bước để kết nối các UD Java với CSDL sử dụng JDBC.
  - Đăng kí the Driver class
  - Tạo connection
  - **Tạo statement**
  - **Thi hành truy vấn queries**
  - **Đóng connection**

## Java Database Connectivity



# Phương thức CreateStatement

- ❖ Để thi hành SQL command, ta sử dụng phương thức *createStatement* của đối tượng Connection.
- ❖ Kết quả trả về là 1 đối tượng statement định nghĩa các phương thức để thi hành câu lệnh SQL.

```
Statement statement = connection.createStatement( );  
  
// execute an UPDATE command  
int count = statement.executeUpdate( "UPDATE City  
    SET population=30000 WHERE name='Bangsaen'" );  
  
System.out.println("Modified " + count + " records");
```

# Statement interface

- ❖ **Statement interface** cung cấp các phương thức để thi hành các câu truy vấn CSDL
- ❖ Các phương thức phổ biến:

<b>public ResultSet executeQuery(String sql)</b>	Thi hành câu truy vấn SELECT. Trả về đối tượng ResultSet
<b>public int executeUpdate(String sql)</b>	Thi hành các truy vấn create, drop, insert, update, delete
<b>public boolean execute(String sql)</b>	Thi hành các truy vấn có thể trả về nhiều kết quả
<b>public int[] executeBatch()</b>	Thi hành 1 batch các mệnh lệnh.

# Statement methods

## ❖ Cách sử dụng các phương thức của Statement

```
ResultSet rs =  
    statement.executeQuery("SELECT ...");  
    ■ use for statements that return data values (SELECT)  
  
int count =  
    statement.executeUpdate("UPDATE ...");  
    ■ use for INSERT, UPDATE, and DELETE  
  
boolean b =  
    statement.execute("DROP TABLE test");  
    ■ use to execute any SQL statement(s)
```

# Statement Example

- ❖ **import** java.sql.\*;
- ❖ **class** FetchRecord{
- ❖ **public static void** main(String args[])**throws** Exception{
- ❖ Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
- ❖ **Statement** stmt=con.createStatement();
- ❖ **int** result=stmt.**executeUpdate**("delete from emp765 where id=33");
- ❖ System.out.println(result+" records affected");
- ❖ con.close();
- ❖ }}



# ResultSet

- ❖ Câu lệnh `statement.executeQuery( )` trả về 1 `ResultSet`.
- ❖ `ResultSet` là bảng chứa kết quả trả về của SQL.

```
Statement statement = connection.createStatement();

// execute a SELECT command
ResultSet rs = statement.executeQuery(
    "SELECT * FROM city WHERE id = "+id );
rs.first(); // scroll to first result
do {
    String name = rs.getString(1);    // get by position
    int population = rs.getInt("population"); // by name
    ...
} while( rs.next() );
```

# ResultSet interface

- ❖ Đối tượng ResultSet duy trì một con trỏ đến 1 hàng của 1 bảng kết quả. Đầu tiên, con trỏ sẽ trỏ đến trước hàng thứ nhất.
- ❖ Các phương thức truy cập phổ biến:

<b>public boolean next():</b>	Di chuyển con trỏ đến hàng kế tiếp
<b>public boolean previous():</b>	Di chuyển con trỏ đến hàng trước hàng hiện tại
<b>public boolean first():</b>	Di chuyển con trỏ đến hàng đầu tiên
<b>public boolean last():</b>	Di chuyển con trỏ đến hàng cuối
<b>public boolean absolute(int row):</b>	Di chuyển con trỏ đến vị trí hàng cụ thể trong bảng kết quả.

# ResultSet Methods

- ❖ `ResultSet` chứa các "row" trả về từ câu query.
- ❖ `ResultSet` hỗ trợ các phương thức để lấy dữ liệu từ cột:
  - "get" by column number -- starts at 1 (not 0)!
  - "get" by column name -- field names in table/query.

```
String query = "SELECT * FROM Country WHERE ...";  
ResultSet rs = statement.executeQuery( query );  
  
// go to first row of results  
rs.first( );  
// display the values  
System.out.println( rs.getString( 1 ) );  
System.out.println( rs.getInt( "population" ) );
```

get by column number



get by name



# ResultSet Methods để lấy dữ liệu

ResultSet "get" trả về dữ liệu cột của hàng đang trỏ tới:

getLong( 3 ) : Lấy dữ liệu theo chỉ số cột (hiệu quả nhất)

getLong( "population" ) : Lấy dữ liệu bằng tên cột (an toàn)

getInt( ), getLong( ) - get Integer field value

getFloat( ), getDouble() - get floating pt. value

getString( ) - get Char or Varchar field value

getDate( ) - get Date or Timestamp field value

getBoolean( ) - get a Bit field value

getBytes( ) - get Binary data

getBigDecimal( ) - get Decimal field as BigDecimal

getBlob( ) - get Binary Large Object

getObject( ) - get any field value

# ResultSet

- ❖ ResultSet gắn kết với 1 statement và 1 connection.
  - Nếu statement or connection bị đóng, kết quả sẽ mất
  - Nếu thi hành câu query khác, kết quả mất
- ❖ ResultSet thay đổi sau khi thi hành câu query
- ❖ ResultSet có thể cập nhật database

```
Statement stmt = connection.createStatement(  
    ResultSet.TYPE_SCROLL_SENSITIVE,  
    ResultSet.CONCUR_UPDATABLE );  
ResultSet rs = statement.executeQuery( query );
```

# ResultSet cập nhật database

- ❖ Xác định thuộc tính `ResultSet.CONCUR_UPDATABLE` khi tạo Statement.
- ❖ Đòi hỏi sự hỗ trợ của database driver

```
// rs is scrollable, will not show changes made
// by others, and will be updatable
Statement statement = connection.createStatement(
    ResultSet.TYPE_SCROLL_INSENSITIVE,
    ResultSet.CONCUR_UPDATABLE );
ResultSet rs = statement.executeQuery(query);
rs.next();
int population = rs.getInt("population");
// add 10,000 to the population
rs.updateInt( "population", population+10000 );
rs.updateRow( );
```

# PreparedStatement

- ❖ Đối tượng PreparedStatement là giao diện phụ của Statement. Được dùng để thực thi các câu lệnh SQL có tham số
- ❖ String sql="insert into emp values(?,?,?)";
- ❖ Ưu điểm: Giúp ứng dụng chạy nhanh hơn do các câu truy vấn có tham số khác nhau chỉ cần biên dịch 1 lần
- ❖

# PreparedStatement set methods

<code>public void setInt(int paramIndex, int value)</code>	Đặt giá trị số nguyên cho tham số có chỉ số tương ứng.
<code>public void setString(int paramIndex, String value)</code>	Đặt giá trị chuỗi cho tham số có chỉ số tương ứng.
<code>public void setFloat(int paramIndex, float value)</code>	Đặt giá trị Float cho tham số có chỉ số tương ứng.
<code>public void setDouble(int paramIndex, double value)</code>	Đặt giá trị Double cho tham số có chỉ số tương ứng.



# Sử dụng PreparedStatement

- ❖ Được tạo ra từ đối tượng Connection.
- ❖ Ví dụ đối tượng PreparedStatement có chứa 2 tham số:
- ❖ `Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");`
- ❖ `PreparedStatement stmt=con.prepareStatement("insert into Emp values(?,?)");`
- ❖ `stmt.setInt(1,101);` //1 specifies the first parameter in the query
- ❖ `stmt.setString(2,"Ratan");`

# Ứng dụng PreparedStatement



```
PreparedStatement updateSales;  
String updateString = "update COFFEES " +  
    "set SALES = ? where COF_NAME like ?";  
updateSales = con.prepareStatement(updateString);  
int [] salesForWeek = {175, 150, 60, 155, 90};  
String [] coffees = {"Colombian", "French_Roast", "Espresso",  
    "Colombian_Decaf", "French_Roast_Decaf"};  
int len = coffees.length;  
for(int i = 0; i < len; i++) {  
    updateSales.setInt(1, salesForWeek[i]);  
    updateSales.setString(2, coffees[i]);  
    updateSales.executeUpdate();  
}
```

# ResultSet Meta-data

- ❖ `ResultSet` có `getMetaData()` trả về các thông tin.
- ❖ `ResultSetMetaData` chứa thông tin miêu tả.

```
try {  
    ResultSet resultSet = statement.executeQuery( query );  
    ResultSetMetaData metadata = resultSet.getMetaData();  
  
    int numberOfColumns = metadata.getColumnCount();  
  
    for(int col=1; col<=numberOfColumns; col++) {  
        // get name and SQL datatype for each column  
        String name = metadata洗getColumnName( col );  
        int type = metadata.getColumnType( col );  
        int typeName = metadata.getColumnTypeName( col );  
    }  
} catch( SQLException sqle ) { ... }
```

# Java DatabaseMetaData interface

- ❖ DatabaseMetaData interface cung cấp các phương thức để lấy meta data về một database như database product name, database product version, tên driver, số tables, tổng số views
- ❖ Sử dụng phương thức getMetaData() của Connection interface để lấy đối tượng của DatabaseMetaData

# Database Metadata

- ❖ `Connection con=DriverManager.getConnection(url,username,password);`
- ❖ `DatabaseMetaData dbmd=con.getMetaData();`
- ❖
- ❖ `System.out.println("Driver Name: "+dbmd.getDriverName());`
- ❖ `System.out.println("Driver Version: "+dbmd.getDriverVersion());`
- ❖ `System.out.println("UserName: "+dbmd.getUserName());`
- ❖ `System.out.println("Database Product Name: "+dbmd.getDatabaseProductName());`
- ❖ `System.out.println("Database Product Version: "+dbmd.getDatabaseProductVersion());`
- ❖ `con.close();`

# Closing the Connection

❖ Khuyến cáo nên đóng connection sau khi hoàn tất

```
Connection connection = DriverManager.getConnection(...);  
/* use the database */  
...  
/* done using database */  
public void close( ) {  
    if ( connection == null ) return;  
    try {  
        connection.close();  
    }  
    catch ( SQLException sqle ) { /* ignore it */ }  
    finally { connection = null; }  
}
```

## Tóm tắt bài học

- ❖ Bài học giới thiệu khái niệm Statement, PreparedStatement, ResultSet để có thể thao tác truy vấn cơ sở dữ liệu đang được kết nối.
- ❖ Ngoài ra ta có thể lấy meta data về dữ liệu trả về từ câu truy vấn Select hoặc từ chính CSDL đang được kết nối.
- ❖ Ta nên đóng kết nối đến CSDL sau khi đã xong tác vụ cần thiết