



VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY
UNIVERSITY OF INFORMATION TECHNOLOGY



Chapter 5

JAVA DATABASE CONNECTIVITY

Lecturer: MSc. Kiet Van Nguyen

Faculty of Information Science and Engineering

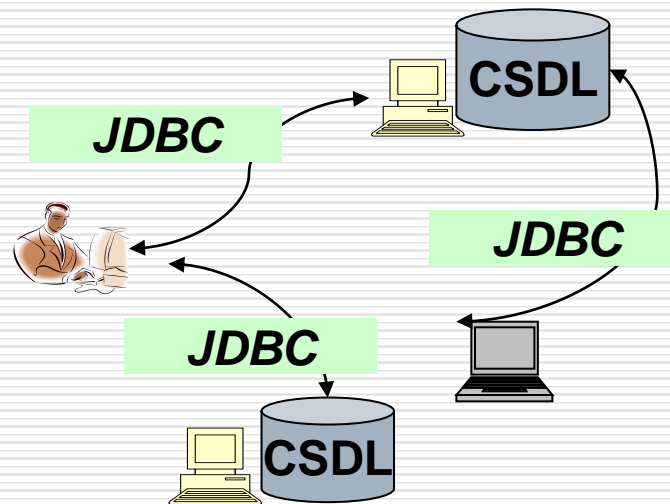
University of Information Technology, VNU-HCM

NỘI DUNG

- ❖ Khái niệm cơ bản
 - ❖ Kiến trúc JDBC & JDBC APIs
 - ❖ Các bước làm việc với Database dùng JDBC
 - ❖ Một số lớp và phương thức cơ bản trong JDBC API
 - ❖ Các loại JDBC Drivers
 - ❖ Ví dụ minh họa
-

Giới thiệu về JDBC

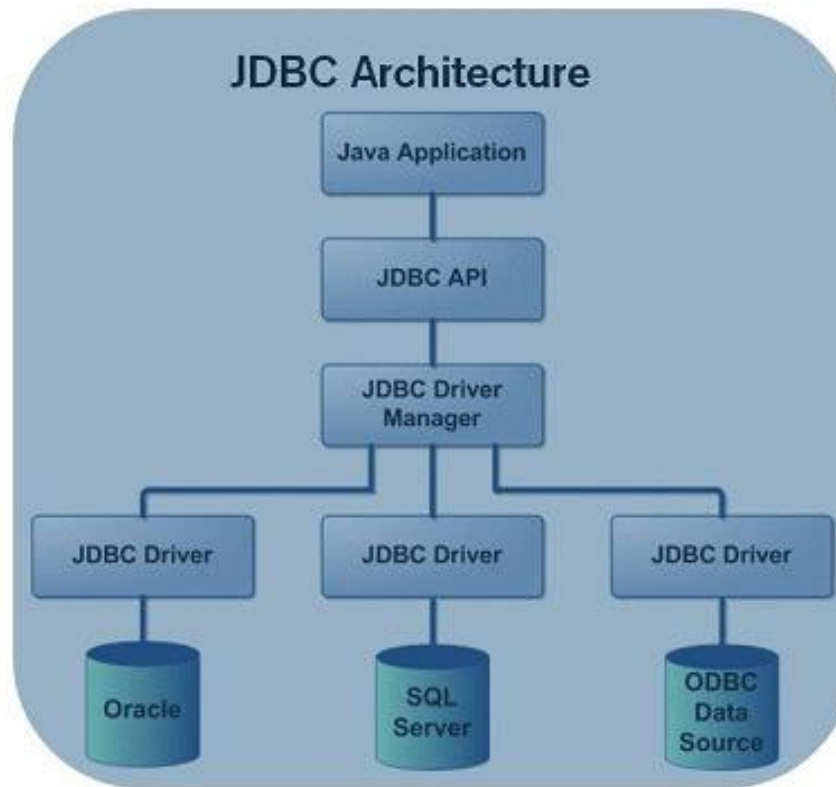
- ❑ JDBC (Java DataBase Connectivity) là một thư viện chuẩn dùng để truy xuất các cơ sở dữ liệu như **MS Access, SQL Server, Oracle,...** trong các ứng dụng Java bằng ngôn ngữ truy vấn SQL.
- ❑ Các hàm truy xuất cơ sở dữ liệu với **JDBC** nằm trong gói `java.sql.*`



Tại sao cần JDBC?

- ❖ JDBC giúp các Java Developers tạo nên các ứng dụng truy xuất cơ sở dữ liệu mà **không cần phải học và sử dụng các APIs do các công ty sản xuất phần mềm khác nhau bên thứ ba cung cấp**. JDBC đảm bảo rằng bạn sẽ có thể phát triển nên các ứng dụng truy cập cơ sở dữ liệu có **khả năng truy cập đến các RDBMS khác nhau** bằng cách sử dụng các JDBC driver khác nhau.
-

Kiến trúc JDBC



Các khái niệm cơ bản

- **JDBC API:** một **API** tiêu chuẩn dùng để tương tác với các loại cơ sở dữ liệu quan hệ. **JDBC** có một tập hợp các class và các Interface dùng cho ứng dụng Java có thể “trò chuyện” với các cơ sở dữ liệu.
 - **JDBC Driver Manager:** Là một class, nó dùng để quản lý danh sách các **Driver** (database drivers).
 - **Các RDBMS** hay các nhà sản xuất phần mềm thứ 3 phát triển các drivers cho java đều phải tuân thủ đặc tả JDBC của SUN.
 - Các java developers dùng các **JDBC drivers** để phát triển các ứng dụng có truy cập và thao tác CSDL.
-

JDBC Drivers

- ❑ Là một chương trình máy tính giúp truy cập đến các hệ DBMS khác nhau dùng kỹ thuật JDBC.
- ❑ Do các hãng xây dựng DBMS hoặc một đơn vị thứ 3 khác cung cấp.

<http://industry.java.sun.com/products/jdbc/drivers>.

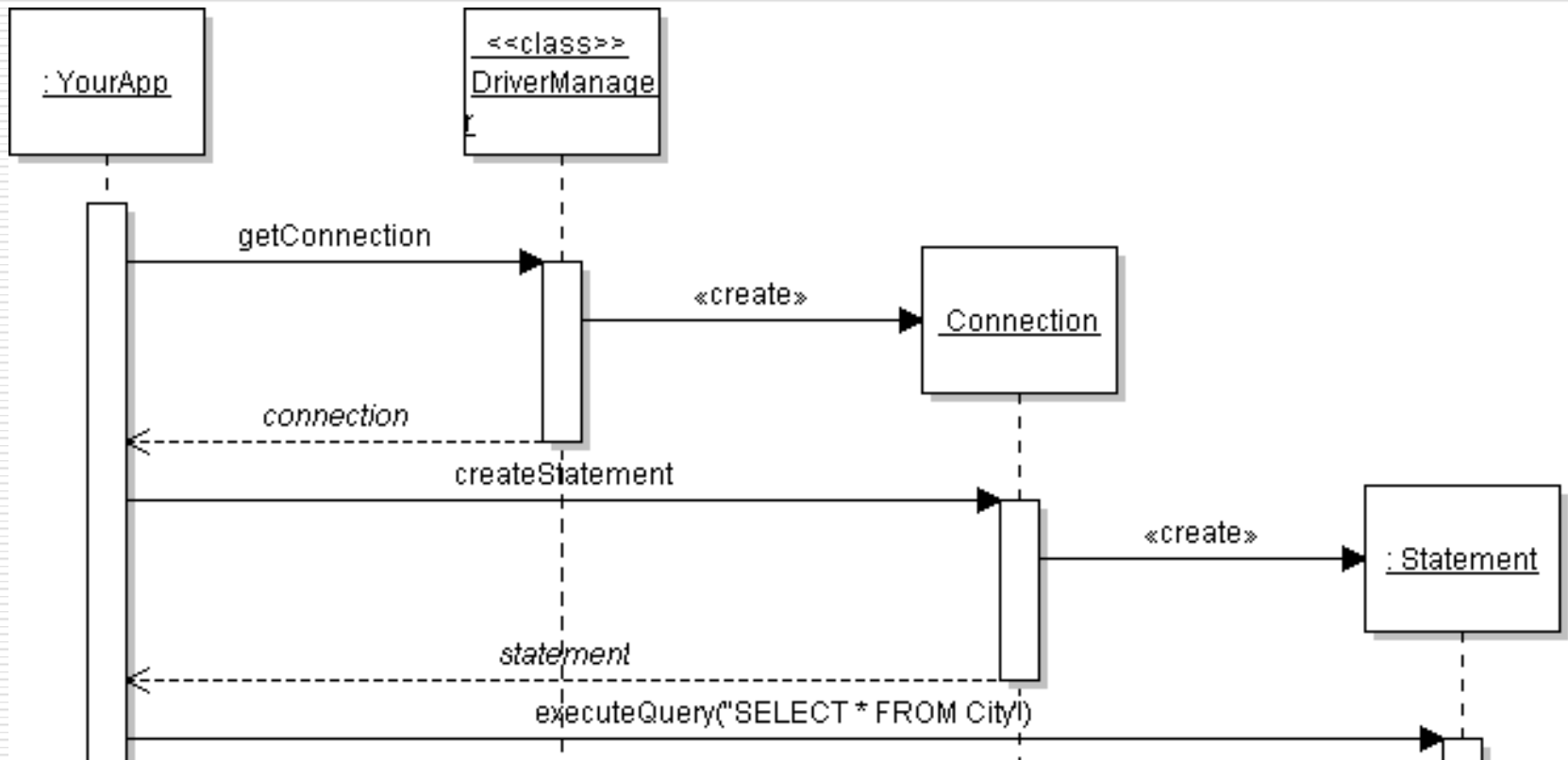
Các loại JDBC Drivers

- ❑ **JDBC-ODBC Bridge**
 - ❑ **A native API** partly Java technology-enabled driver
 - ❑ **Pure Java Driver** for Database Middleware
 - ❑ **Direct-to-Database** Pure Java Driver
-

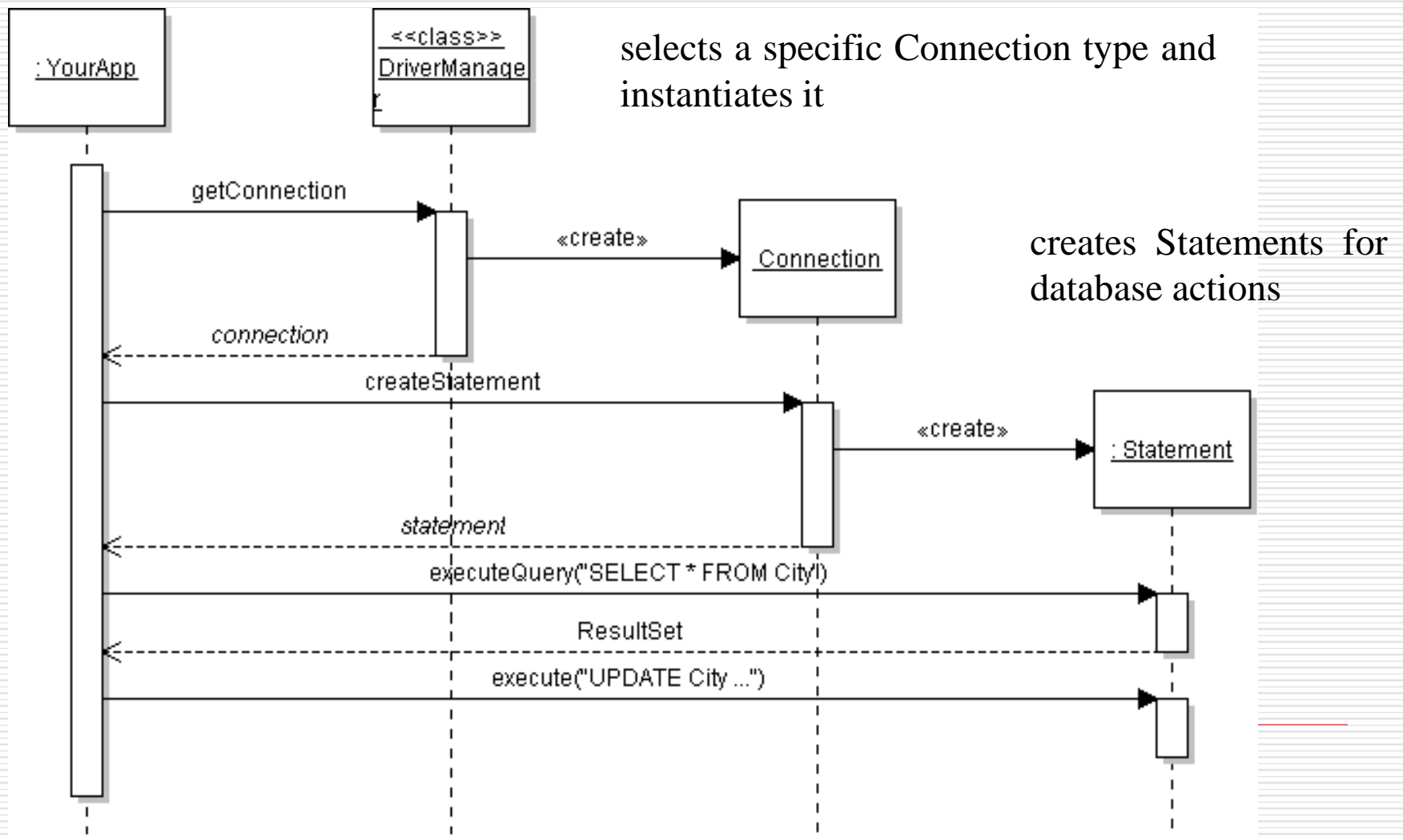
Các bước thực hiện thao tác với CSDL

- ❖ **Bước 1**: Nạp JDBC driver.
 - ❖ **Bước 2**: Tạo kết nối với CSDL dùng driver đã nạp ở **Bước 1**.
 - ❖ **Bước 3**: Thao tác với CSDL: thêm, xoá, sửa, cập nhật, ...
-

Các bước thực hiện thao tác với CSDL



Các bước làm việc với CSDL



Một số lớp và phương thức cơ bản

- ❖ **DriverManager**: Nạp các JDBC driver vào trong bộ nhớ. Có thể sử dụng nó để mở các kết nối tới một nguồn dữ liệu.
 - ❖ **Connection** : Biểu thị một kết nối đến một nguồn dữ liệu. Được dùng để tạo ra các đối tượng Statement, PreparedStatement,
 - ❖ **Statement**: Biểu diễn một lệnh SQL tĩnh. Kết quả trả về là đối tượng ResultSet.
 - ❖ **PreparedStatement**: Một giải pháp thay thế hoạt động tốt hơn đối tượng Statement, thực thi một câu lệnh SQL đã được biên dịch trước.
-

JDBC Code

```
static final String URL = "jdbc:mysql://dbserver/world";
static final String USER = "student";
static final String PASSWORD = "secret";

//Load driver
Class.forName("JDBC-DriverName");

// 1. Get a Connection to the database.
Connection connection =
    DriverManager.getConnection( URL, USER, PASSWORD );

// 2. Create a Statement
Statement statement = connection.createStatement();

// 3. Execute the Statement with SQL command.
ResultSet rs = statement.executeQuery("SELECT * FROM ...");

// 4. Use the Result.
while ( rs.next( ) ) {
    String name = rs.getString("name");
```

Kết nối CSDL với JDBC

- ❑ **java.sql.Connection** là một interface chuẩn để kết nối đến các HQTCSDL.
 - ❑ **DriverManager** dùng để chọn driver và thiết lập kết nối.
-

JDBC URL

- Chỉ định nguồn dữ liệu sẽ kết nối
jdbc:<subprotocol>:<dsn>:<others>

Trong đó:

- **<subprotocol>**: được dùng để xác định trình điều khiển để kết nối với CSDL.
- **<dsn>**: địa chỉ CSDL. Cú pháp của **<dsn>** phụ thuộc vào từng trình điều khiển cụ thể.
- **<others>**: các tham số khác

Ví dụ:

- **jdbc:microsoft:sqlserver://hostname:1433** là URL để kết nối với CSDL Microsoft SQL Server. Trong đó hostname là tên máy cài SQL Server.
-

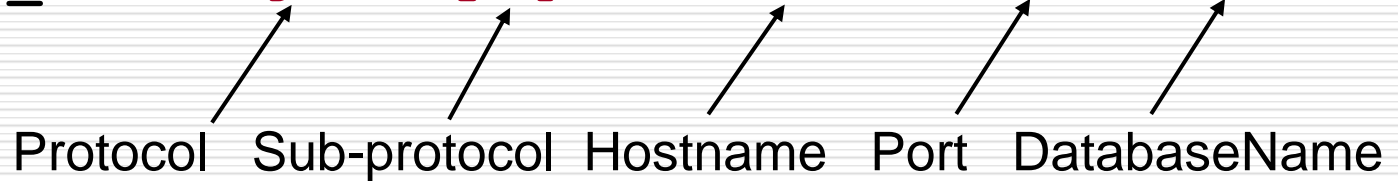
Các JDBC URL phổ biến

RDBMS	Database URL format
MySQL	<code>jdbc:mysql://hostname:portNumber/databaseName</code>
ORACLE	<code>jdbc:oracle:thin:@hostname:portNumber:databaseName</code>
DB2	<code>jdbc:db2:hostname:portNumber/databaseName</code>
Java DB/Apache Derby	<code>jdbc:derby:databaseName</code> (embedded) <code>jdbc:derby://hostname:portNumber/databaseName</code> (network)
Microsoft SQL Server	<code>jdbc:sqlserver://hostname:portNumber;databaseName=databaseName</code>
Sybase	<code>jdbc:sybase:Tds:hostname:portNumber/databaseName</code>

Database URL

Định dạng chung của database URL:

```
String DB_URL = "jdbc:mysql://dbserver:3306/world";
```



Protocol Sub-protocol Hostname Port DatabaseName

- Port là TCP port mà hệ QTCSDL sử dụng để lắng nghe yêu cầu.
 - **3306** is the **default port** for MySQL
- Sử dụng "localhost" nếu CSDL nằm cùng 1 máy.

Database URL

hostname và port là tùy chọn.

Đối với MySQL driver: **defaults** là localhost và port 3306

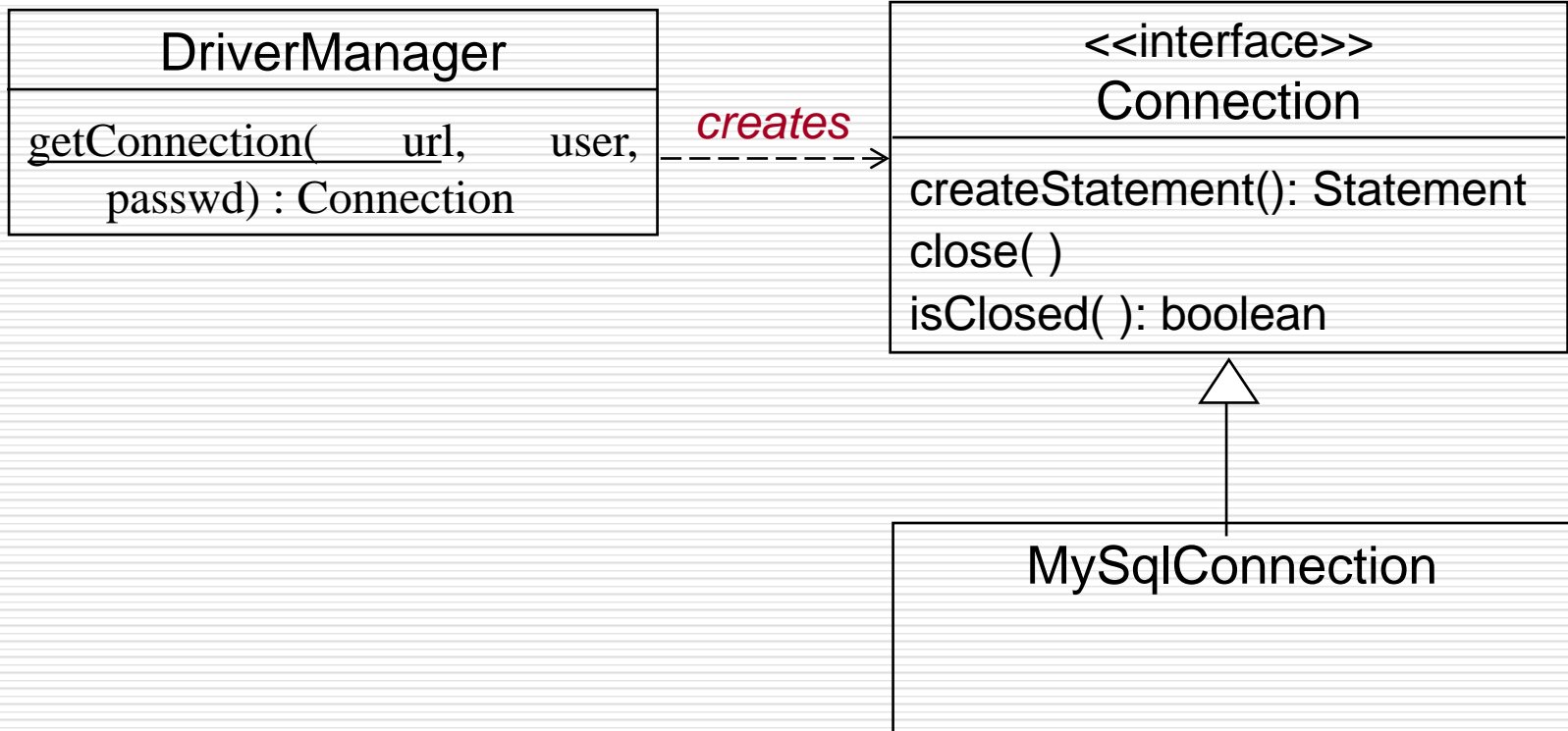
Ví dụ:

```
"jdbc:mysql://localhost:3306/world"
```

```
"jdbc:mysql://localhost/world"
```

Kết nối CSDL với JDBC

url = "jdbc:mysql://hostname/database"



Thi hành SQL Commands

- ❑ Để thi hành SQL command sử dụng pt `createStatement` của đối tượng `Connection`.
- ❑ **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");
```

Thi hành câu lệnh SQL

- ❑ 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() );
```

Ví dụ

```
Scanner console = new Scanner(System.in);  
System.out.print("Name of city to find? ");  
String name = console.nextLine().trim();  
String query =  
    "SELECT * FROM city WHERE Name=" +  
        name + " ";  
ResultSet rs =  
    statement.executeQuery( query );
```

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 index -- 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 index



get by name



ResultSet Methods

- ❑ ResultSet hỗ trợ các phương thức để lấy từng dòng và cột trong kết quả trả về

ResultSet
<code>next() : boolean</code>
<code>previous() : boolean</code>
<code>first() : boolean</code>
<code>last() : boolean</code>
<code>absolute(k)</code>
<code>getInt(name: String)</code>
<code>getInt(index: int)</code>
<code>...</code>

ResultSet Methods for Getting Data

ResultSet "get" methods return column data:

getLong(3) : get by column index (most efficient)

getLong("population") : get by field name (safest)

```
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
getObject( )              - get any field value
...
```

Thi hành câu lệnh SQL Commands

Statement interface định nghĩa nhiều phương thức

```
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)

Các bước làm việc với CSDL

❖ Thao tác với CSDL:

```
Statement stmt = conn.createStatement();
try {
    ResultSet rs = stmt.executeQuery( "SELECT * FROM MyTable" );
    try {
        while ( rs.next() ) {
            int numColumns = rs.getMetaData().getColumnCount();
            for ( int i = 1 ; i <= numColumns ; i++ ) {
                // Column numbers start at 1.
                // Also there are many methods on the result set to return
                // the column as a particular type. Refer to the Sun documentation
                // for the list of valid conversions.
                System.out.println( "COLUMN " + i + " = " + rs.getObject(i) );
            }
        }
    } finally {
        rs.close();
    }
} finally {
    stmt.close();
}
```

Ví dụ minh họa – JDBC ODBC

...

Connection myCon;

Statement myStatement;

ResultSet myResultSet;

String sUsername, sPassword;

try {

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

myCon = **DriverManager.getConnection**("jdbc:odbc:ThuchanhJ2EE", "", "");

myStatement = myCon.**createStatement**();

myResultSet = myStatement.**executeQuery**("Select * from Account");

Ví dụ minh họa - JDBC ODBC

```
while (myResultSet.next()) {  
    sUsername = myResultSet.getString(1);  
    sPassword = myResultSet.getString(2);  
    if (sUsername.equals("admin") && sPassword.equals("admin"))  
        return true;  
}  
  
myResultSet.close(); myStatement.close();    myCon.close();  
  
}  
  
catch(Exception e) {  
    System.out.println(e.toString());  
}
```

PreparedStatement

- ❑ Đối tượng PreparedStatement chứa 1 câu lệnh SQL đã được biên dịch trước.
 - Khi thực thi DBMS không cần phải biên dịch câu lệnh SQL.
- ❑ Thường được dùng với các câu lệnh SQL có tham số.

PreparedStatement

- ❑ Được tạo ra từ đối tượng Connection.
- ❑ Ví dụ đối tượng PreparedStatement có chứa 2 tham số:

“SELECT lastName, firstName, title ” +
“FROM authors, titles, authorISBN ” +
“WHERE authors.authorID = authorISBN.authorID ” +
“AND titles.ISBN = authorISBN.isbn AND ” +
“lastName = ? AND firstName = ?”);

Cung cấp giá trị cho tham số của PreparedStatement

- ❑ Trước khi thi hành, chúng ta cần cung cấp giá trị cho tham số trong đối tượng PreparedStatement.
- ❑ Thực hiện thông qua các phương thức setXXX.
 - `authorBooks.setString(1, “Deitel”);`
 - `authorBooks.setString(2, “Paul”);`

Ví dụ

```
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

- ❑ 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

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

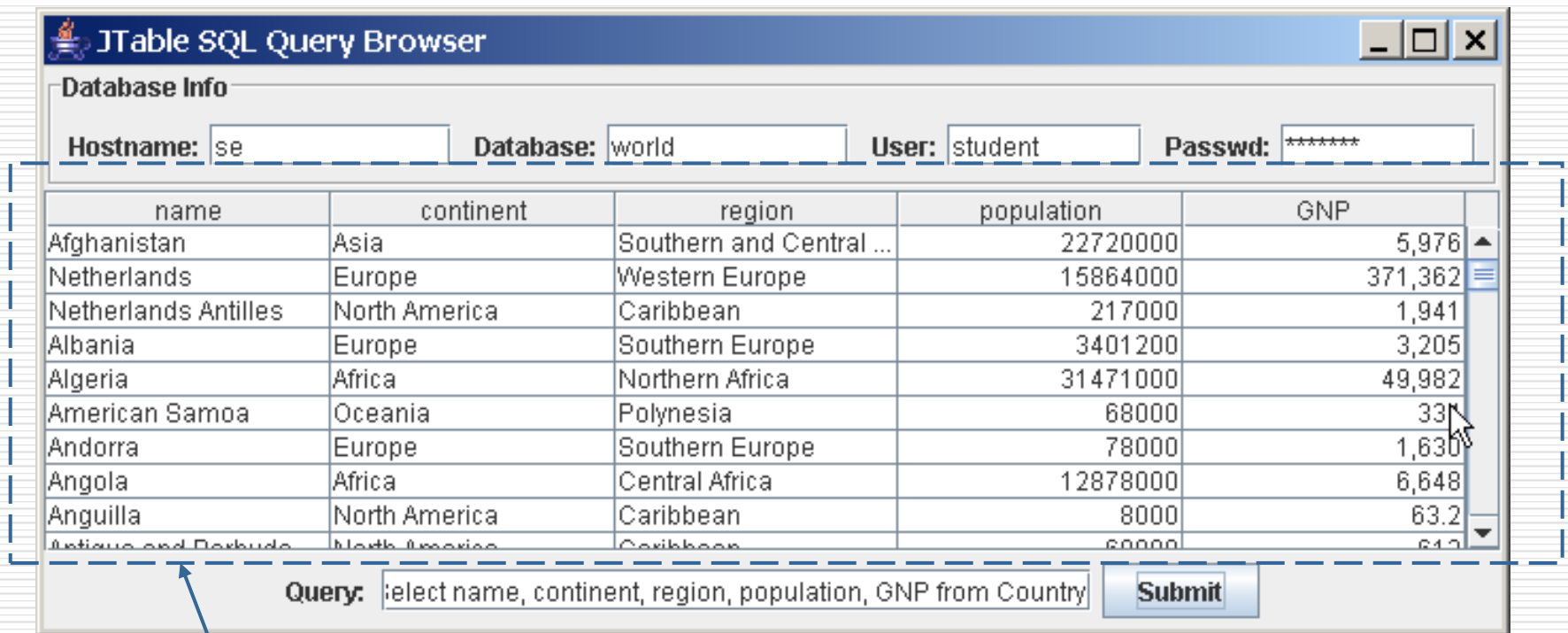
ResultSet cập nhật database

- ❑ 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( );
```

JTable

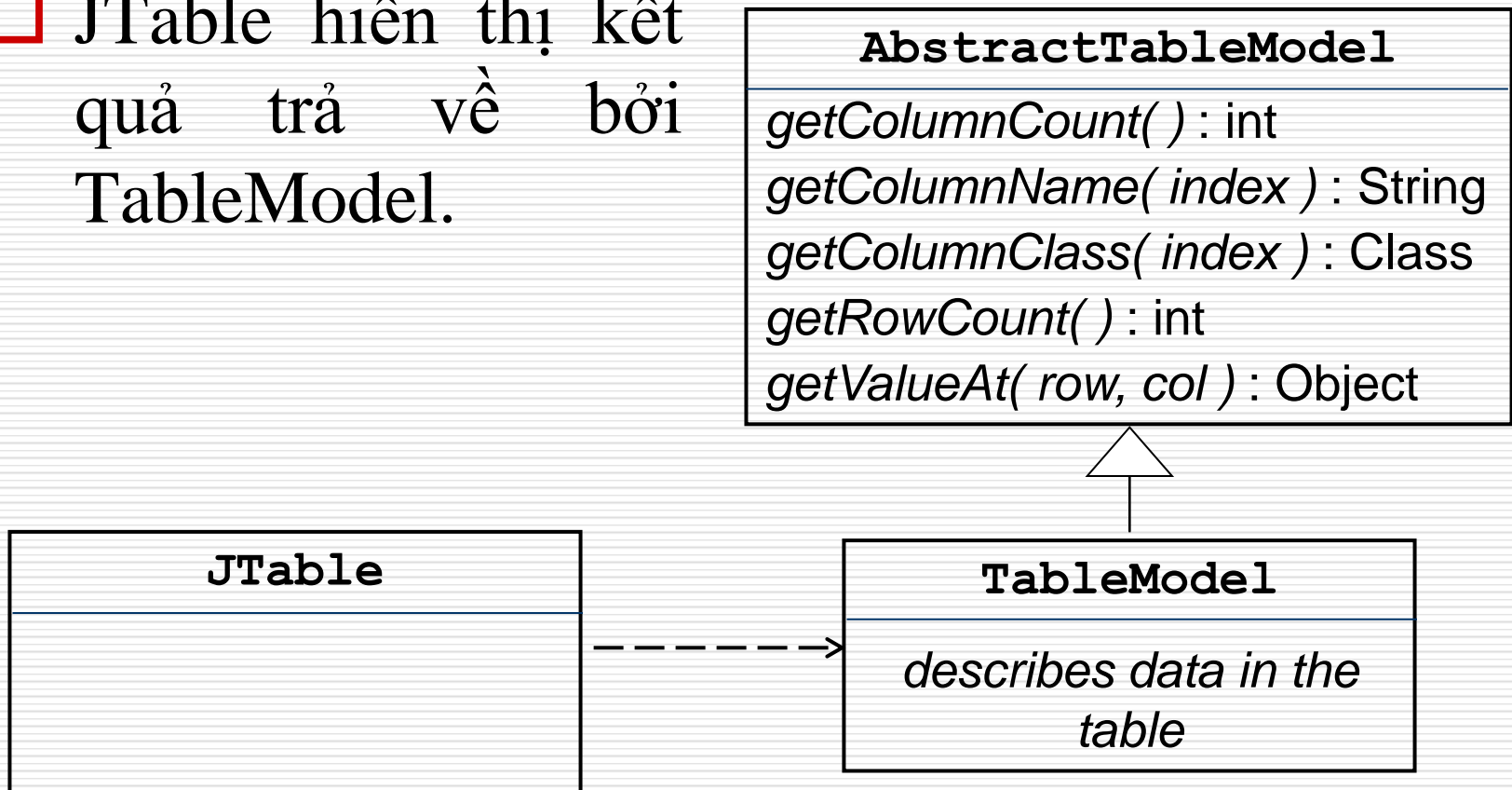
- ❑ Swing object hiển thị dữ liệu dưới dạng bảng.



A JTable

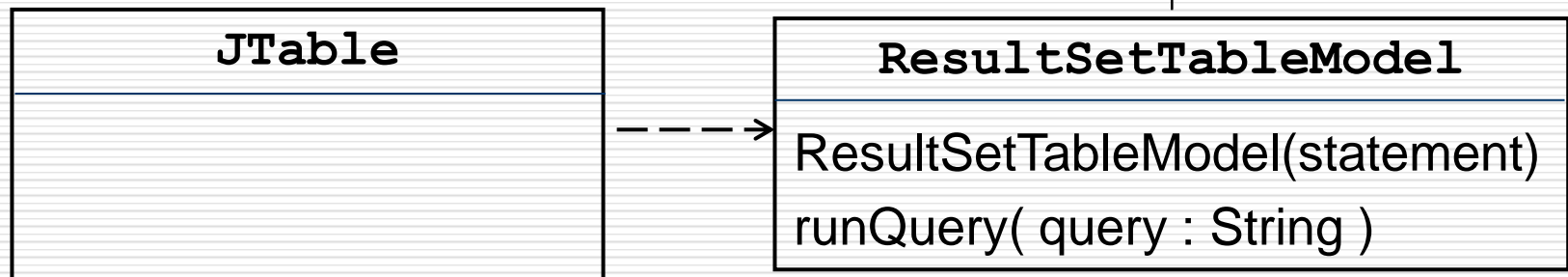
JTable Class Diagram

- ❑ JTable hiển thị kết quả trả về bởi TableModel.



Design a TableModel for Queries

- Design a TableModel to manage a ResultSet



Cài đặt TableModel

❑ ResultSet chưa dữ liệu cần hiển thị.

```
class ResultSetTableModel extends AbstractTableModel {
    private Statement statement;
    private ResultSet rs;

    public Object getValueAt(int row, int col) {
        if ( rs == null ) return null;
        rs.absolute( row + 1 );
        rs.getObject( col );
    }

    public int getRowCount() {
        if ( rs == null ) return 0;
        rs.last(); // move to last row
        rowCount = rs.getRow();
        return rowCount;
    }
}
```

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; }  
}
```


Q&A

Thank you!