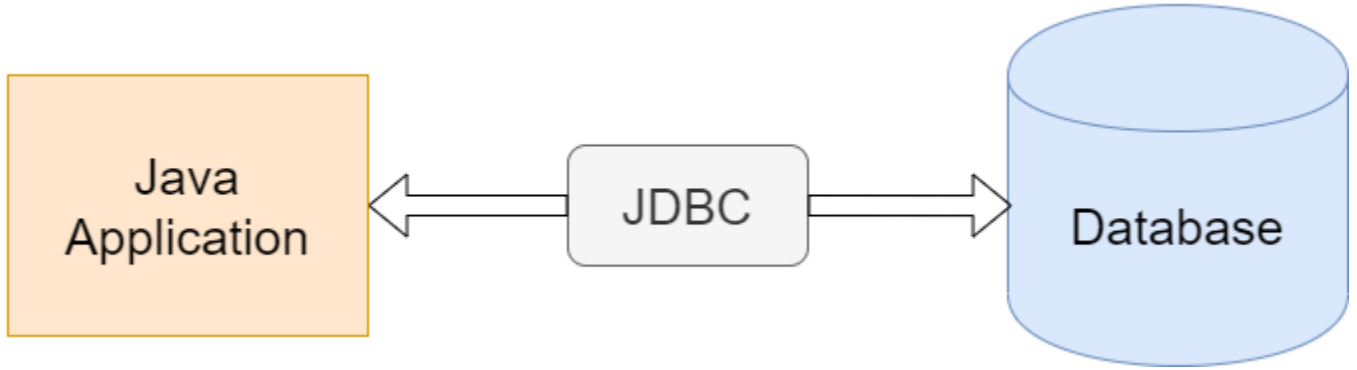
Chương 5: Truy xuất Database với JDBC

Trường Đại học Công nghệ Sài gòn
Khoa Công nghệ Thông tin

1. Tổng quan JDBC

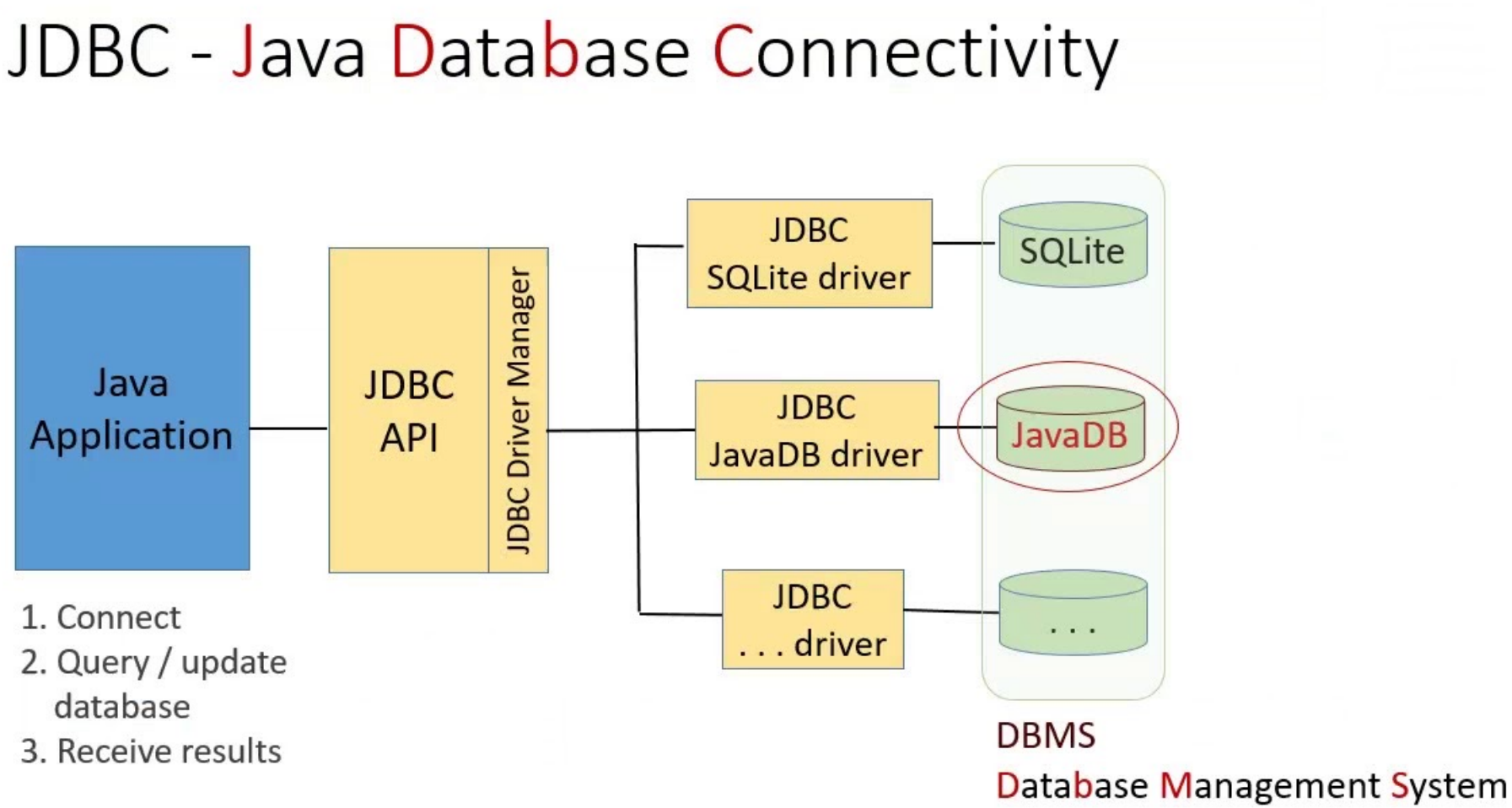
- JDBC là chuẩn kết nối CSDL, cung cấp các **API** cho phép các ứng dụng Java tương tác với các hệ quản trị CSDL
- JDBC version
  - JDBC 4.1->J2SE 7.0
  - JDBC 4.2->J2SE 8.0



2. Kiến trúc JDBC

- **DriverManager**
  - Là một java class trong package java.sql
  - Quản lý các database driver
  - Sử dụng đăng ký các database driver
  - Connection con=**DriverManager**.getConnection(jdbcurl,user,pass)
- **Database Driver**
  - Hoạt động như một cầu nối giữa java app và database

2. Kiến trúc JDBC



3. JDBC API

- Gồm 2 gói thư viện:
  - java.sql.*: truy cập và xử lý dữ liệu được lưu trữ trong CSDL bằng java.
  - javax.sql.*: là trình điều khiển JDBC (**có 4 kiểu jdbc driver**).
- Các lớp và giao diện chính:

3. JDBC API

Interface	Classes
Driver	DriverManager
Connection	Date
Statement	Time
PreparedStatement	TimeStamp
CallableStatement	Types
ResultSet	
ResultSetMetaData	
DataBaseMetaData	

4. Các kiểu JDBC Driver

- Type 1: JDBC/ODBC:
  - Truy xuất bất kỳ DBMS có hỗ trợ ODBC
  - Khả chuyển cao, chậm.
  - Chỉ hỗ trợ dưới jdk 17
- Type 2:Native API
  - JDBC tương tác trực tiếp với Database
  - 1 phần mã Java
  - VD Oracle Client, not MySQL

4. Các kiểu JDBC Driver

- Type 1: JDBC/ODBC:
  - Truy xuất bất kỳ DBMS có hỗ trợ ODBC
  - Khả chuyển cao, chậm.
  - Chỉ hỗ trợ dưới jdk 17
- Type 2:Native API
  - JDBC tương tác trực tiếp với Database
  - 1 phần mã Java
  - VD Oracle Client, not MySQL

4. Các kiểu JDBC Driver

Nên sử dụng dạng nào???

- Nếu sử dụng 1 dạng database -> type 4
- Nếu sử dụng nhiều dạng database-> type 3
- Nếu type 3, 4 không hỗ trợ→ type 2
- Cuối cùng chọn type 1

Cài đặt như thế nào???

- Tải jdbc driver tương ứng CSDL.
- Thêm vào classpath của ứng dụng.

5. Các bước cơ bản trong JDBC

1. Import gói JDBC
2. Mở kết nối với cơ sở dữ liệu.
3. Tạo một đối tượng câu lệnh để thực hiện truy vấn.
4. Thực thi đối tượng câu lệnh và trả về một tập kết quả truy vấn.
5. Xử lý bộ kết quả.

Lưu ý:

- Từ JDBC 4.0+, không cần "Class.forName()", driver được tìm thấy trong classpath được tự động nạp.
- Sử dụng **try-with-resource** tự đóng tài nguyên JDBC.

6. JDBC Code

```
//1
import java.sql.*;

//2
try (Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/jdbc_lab","root", ""))

//3
Statement stmt = con.createStatement();

//4
ResultSet rs = stmt.executeQuery("select * from lop");

//5
while(rs.next()) {
        System.out.println(rs.getInt(1) + "\t" + rs.getString(2));
}
```

(1) kết nối Database

- **Thư viện: java.sql.Connection**
- **Thư viện jdbc driver**.
  - MySQL driver

    `mysql-connector-java-5.1.7-bin.jar`
  - Derby driver: `derby.jar` **hoặc** `derbyclient.jar`
  - HSQLDB driver: `hsqldb.jar`
- **DriverManager :** chọn driver dựa trên URL.

Database Driver?

Tích hợp Driver vào ứng dụng:

Driver được đóng dưới dạng JAR file.

Chép JAR file theo 4 cách:

1. Thêm vào IDE project: *external jar file*

2. Thêm tới CLASSPATH

CLASSPATH = /my/path/mysql-connector.jar;.

Database Driver?

Tích hợp Driver vào ứng dụng:

Driver được đóng dưới dạng JAR file.

Chép JAR file theo 4 cách:

3. Thêm JAR sử dụng command line:

```
java -cp /my/path/mysql-connector.jar ...
```

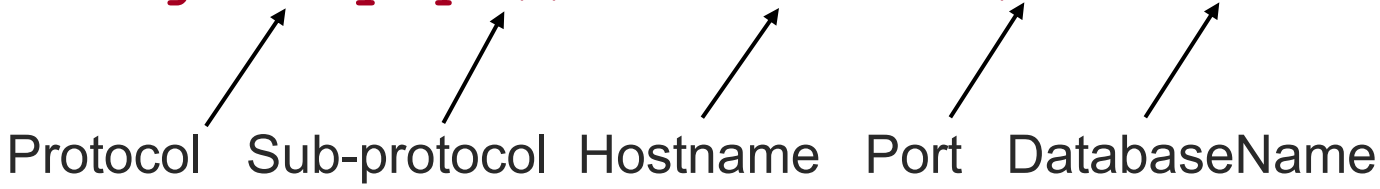
4. Chép JAR file vào thư mục : **JRE/lib/ext** :

```
C:/java/jre1.6.0/lib/ext/mysql-connector.jar
```

Database URL

Cấu trúc URL : Tùy thuộc DBMS sử dụng

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



- **Tiếng việt:**
  ?useUnicode=true&characterEncoding=UTF-8

Oracle thin driver:
String URL="jdbc:oracle:thin:@localhost:1521:XE"

Method getConnection(String url,Strin user,String password)

Câu lệnh Statement

- Các loại Statement
  - Statement: thi hành câu lệnh tùy ý tại thời điểm chạy
  - PreparedStatement: câu lệnh SQL được biên dịch trước
  - CallableStatement: gọi thủ tục trên DBMS

Câu lệnh Statement

- Sử dụng kết nối tạo câu lệnh
  - Statement s = con.createStatement()
  - PreparedStatement s = con. prepareStatement(String)
  - CallableStatement s = con. prepareCall(String)

Statement

- Các phương thức chính
  - public **ResultSet** **executeQuery**(String sql)
    - Sử dụng cho các câu lệnh select..
  - public int executeUpdate(String sql)
    - Sử dụng cho các câu lệnh insert , delete, update…

Statement

- Các phương thức chính
  - public Boolean execute(String sql)
    - Sử dụng cho các câu lệnh tùy ý
  - public int[] executeBatch()
    - Sử dụng cho bó lệnh
    - Thêm lệnh: stm.addBatch(String sql)

PreparedStatement

- Tăng hiệu quả thi hành câu lệnh SQL.
- Biên dịch một lần và gọi thi hành nhiều lần.
- Thay đổi đối số mỗi lần thi hành-> thích hợp với các câu lệnh có đối số

```
PreparedStatement updateAddr=con.preparedStatement(
"UPDATE Customers SET Address=? WHERE CustNo=?");
updateAddr.setString(1,"Danang");
updateSales.setInt(2,1001);
```

CallableStatement

- Thi hành các thủ tục đã cài đặt sẵn trên DBMS
  - Oracle→PL/SQL
  - MYSQL→Stored procedure Language
  - SQL server→Transact SQL(TSQL)
- Cú pháp:
  - {Call procedureName(a1,a2,…)};

CallableStatement

- Các đối số có thể là IN, OUT hoặc cả 2 INOUT
- CallableStatement cst=con.prepareCall("{call pro(?,?)}");
- Truyền đối số IN bằng hàm setxxx();
- Đăng kí đối số OUT trước khi thi hành thủ tục
  - registerOutParameter(1,Types,VARCHAR)
- Đối số IN OUT

Ví dụ: CallableStatement

```
Create procedure addProc(num1 IN number,num2 IN
number, num3 OUT number)
BEGIN
        num3:=num1+num2;
END;
```

```
Connection  con =
DriverManager.getConnecton(……..);
CallableStatement cst=con.prepareCall(“call
addProc(?,?,?)”);
cst.setInt(1,100);
cst.setInt(2,200);
cst.registerOutParameter(2,Types.INTEGER);
cst.execute();
System.out.println(cst.getInt(3);
con.close();
```

Thực thi Statement

- Có 3 cách
  - ResultSet executeQuery():
    - Thi hành các câu lệnh truy vấn
    - Trả về kết quả qua đối tượng ResultSet
    - ResultSet rs=s.executeStatement(“select from …..”);

Thực thi Statement

- Có 3 cách
  - int executeUpdate()
    - Dùng cho câu lệnh cập nhật dữ liệu
    - Trả về số record bị ảnh hưởng

Thực thi Statement

- Có 3 cách
  - boolen execute()
    - Khi không biết rõ câu lệnh là truy vấn hay cập nhật
    - Trả về true nếu câu lệnh là truy vấn
      - Gọi getResultSet() nhận kết quả truy vấn truy vấn
    - Trả về false nếu câu lệnh là cập nhật
      - Gọi getUpdateCount() số record được cập nhật

ResultSet

- Được chia làm 2 dạng:
  - Read only ResultSet (Mặc định)
  - Updatable ResultSet
- Một **ResultSet** chứa một "row" cho mỗi kết quả trả về (bắt đầu từ 1).

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

```
getInt( ), getLong( )
getFloat( ),getDouble()
getString( )
getDate( )
getBoolean( )
getBytes( )
getBigDecimal( )
getBlob( )
getObject( )
```

ResultSet Methods

```
String query = "SELECT * FROM Lop WHERE ...";
ResultSet rs = statement.executeQuery( query );

rs.first( );
// Hiển thị kết quả
System.out.println( rs.getString( 1 ) );
System.out.println( rs.getString( "Tenlop" ) );
```

column number

Column name

```
//Duyệt danh sách kết quả
while ( rs.next( ) ) {
   String id = rs.getString("Lopid");
   String tenlop = rs.getString("Tenlop");
   System.out.println( id +" "+tenlop );
}
```

Quản lý Transaction

- Có 2 dạng transaction
  - Local: Cùng 1 database
  - Global: Trên nhiều database
- JDBC chỉ hỗ trợ local, global phải dùng EJB hoặc spring framework

Quản lý Transaction

- Ngăn cấm chế độ autocommit

  con.setAutoCommit(false)

- Khi hoàn tất các tác vụ commit transaction sử dụng

  con.commit()

- Nếu có lỗi khi thực thi lệnh gọi phương thức phục hồi

  con.rollback()

CaseStudy 1

- Database Mysql: 2020
- Viết chương trình xuất ra danh sách customer


```
Customer{id=1, name=Hồ Đình Khả, age=52}
Customer{id=2, name=Lương An Vinh, age=30}
Customer{id=3, name=Lê Triệu Ngọc Đức, age=35}
BUILD SUCCESSFUL (total time: 2 seconds)
```

Các bước thực hiện

- Tạo Project
- Nạp Thư viện
- Viết chương trình
  - (1) Nạp Driver, định nghĩa kết nối
  - (2)Khởi tạo Statement
  - (2)Thực thi Query
  - (2) Xử lý kết quả

Tạo một Class quản lí kết nối

Tạo class ConnectionManager với phương thức static

ConnectionManager
- connection : Connection
+getConnection( ) : Connection
+close( ) : void

```
// Sử dụng class ConnectionManager
Connecton con=ConnectionManager.getConnection();
Statement statement = con.createStatement( );
```

Class ConnectionManager

```
public class ConnectionManager {
 private static String driver = "com.mysql.jdbc.Driver";
 private static String url = "jdbc:mysql://hostname/world";
 private static String user = "student";
 private static String pwd = "student";
 private static Connection connection = null;
 private ConnectionManager() {}
 public static Connection getConnection( ) throws ... {
   if ( connection == null ) connection = makeConnection();
   return connection; }
```

Class ConnectionManager

```
private static Connection makeConnection( )
                                throws SQLException {
   try {
      Class.forName( driver );
      connection = DriverManager.getConnection(
                  url, user, pwd );
  } catch ( FileNotFoundException ex ) {
     System.out.println("Loi ket noi");
   }
    return connecton;
}
```

Sử dụng tập tin thuộc tính Properties

- db.properties

```
driver = "com.mysql.jdbc.Driver"
url = "jdbc:mysql://hostname/world"
user = "student"
pwd = "student"
```

Sử dụng tập tin thuộc tính Properties

- Sử dụng trong code

```
Proprties p=new Properties();
FileInputStream fis=new
FileInputStream("db.properties");
p.load(fis);
String driver =p.getProperty("driver");
String url =p.getProperty("url");
String user=p.getProperty("user");
Sting pwd=p.getProperty("pwd");
```

Tạo đối tượng truy xuất database (Dao)

CustomerDao
findAll():List<Customer> findByid(id: String): Customer insert(c:Customer) : boolean delete(id:int) : boolean update(c:Customer):boolean findByName(name: String): List<Customer>

CustomerDao

```
public class CustomerDao {
   public List<Customer> findAll() {}
    public Customer findByid(String id ) {}
   public boolean insert(Customer c) {}
   public boolean delete(int id) {}
   public boolean update(Customer l) {}
    public List<Customer> findByName( String name ) {}
}
```

CustomerDao

```
public <.return type>  nameMethod(parameter….) {
        try{
            Connection con=ConnectionManaget.getConection();
             String sql="    ";  //câu lệnh truy vấn
                PreparedStatement ps=con.prepareStatement("sql");
             //Cung cấp tham số câu truy vấn nếu có ps.setType(index, value);
             //Thực thi câu lệnh : ps.executeQuery() hoặc ps.executeUpdate()
            //Xử lý kết quả
            return ……; //
        }catch(Exception e){
                System.out.println("Thông báo lỗi truy cập");
        }
           return …….;
}
```

- **Lưu ý:**
  - Kiểu dữ liệu Date khi lưu xuống database cần chuyển đổi
    - java.uti.Date→java.sql.Date
    - java.sql.Date dsql=new java.sql.Date(duti.getTime());
  - Đối với khóa tự động: khi thêm mới cần truy xuất id do RDBMS tạo ra

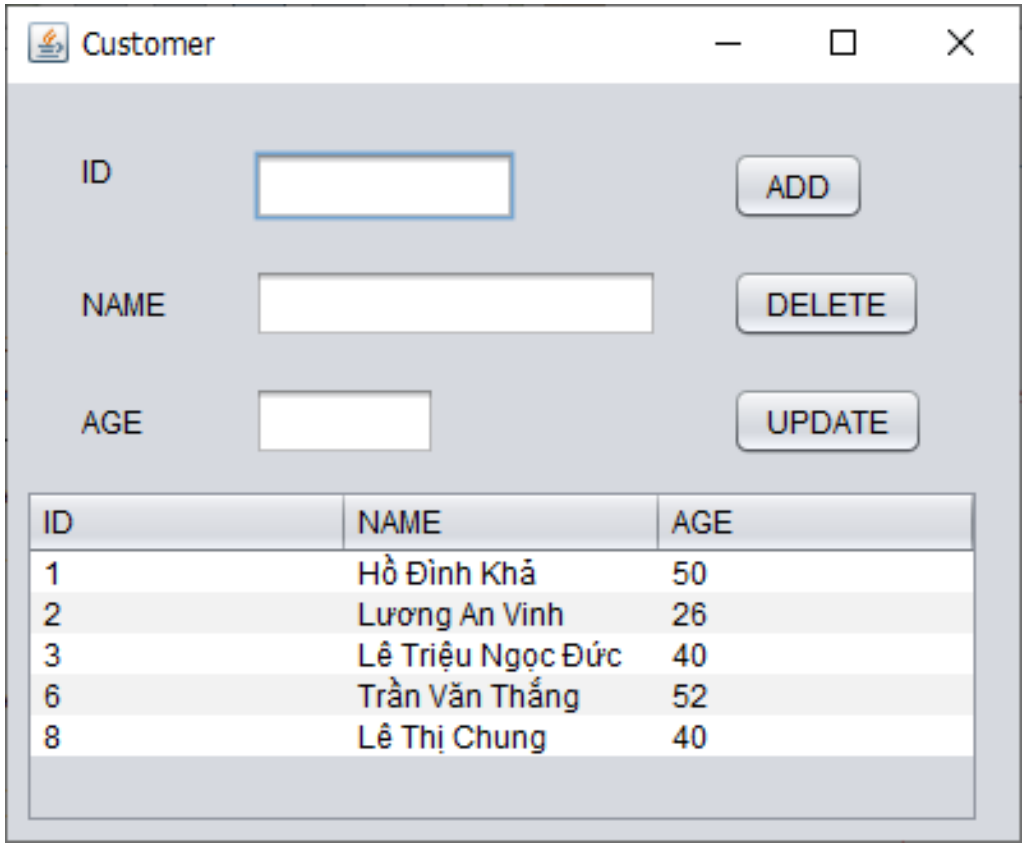
CustomerDao

```
public boolean insert(Customer c) {
    try {
        Connection con = ConnectionManager.getConnection();
        PreparedStatement ps = con.prepareStatement("insert into customer(name,age) values(?,?)",
                PreparedStatement.RETURN_GENERATED_KEYS);
        ps.setString(1, c.getName());
        ps.setInt(2, c.getAge());
        int i = ps.executeUpdate();
        if (i != 0) {
            ResultSet key = ps.getGeneratedKeys();
            if (key.next()) {
                int cusid = (int) key.getInt(1);
                c.setId(cusid);
                return true;
            }
        }
    } catch (Exception e) {
        System.out.println("Loi Them moi");
    }
    return false;
}
```

Bỏ nếu id không auto

Bài tập

- Hoàn tất code cho CustomerDao
- Viết ứng dụng hoàn chỉnh.



Tổng kết

- JDBC là một chuẩn giao tiếp cho phép truyền thông với các dạng database khác nhau
- Để sử dụng JDBC , cần một driver JDBC hoặc ODBC của database tương ứng.
- Chương trình cần nạp một driver. DriverManager sẽ chọn một driver khi khởi tạo một Connection.
- Một đối tượng Connection : quản lí nối kết tới database.

Tổng kết

- Một đối tượng Statement được dùng để thực hiện các lệnh tương tác database và trả về kết quả.
- Một truy vấn trả về một ResultSet chứa :data và meta-data.
- Một ResultSet có thể "read-only" hoặc "updateable" dựa vào đối tượng Statement.
- Cần đóng Statement hoặc Connection khi hoàn thành: thu hồi tài nguyên và đảm bảo tính toàn vẹn.

Resources

MySQL

- http://dev.mysql.com/

Learning SQL

- http://www.w3schools.com/sql/
  nice tutorial and command reference

Learning JDBC

- JDBC Trail in Sun's *Java Tutorial*.
- Dietel, *Java How To Program*, Chapter 25.
- ... and *zillions* of resources on the web