



cep.edu.vn

TRƯỜNG CAO ĐẲNG KINH TẾ - KẾ HOẠCH ĐÀ NẴNG

KHOA TIN HỌC & NGOẠI NGỮ

IFF

SLIDE BÀI GIẢNG

LẬP TRÌNH JAVA 2

THS. NGUYỄN ĐÌNH THÀ

Đà Nẵng – 2022

HƯỚNG DẪN CÀI ĐẶT CÁC ỨNG DỤNG CHO MÔN HỌC

1. Cài đặt trên Windows

Tải và cài đặt:

Netbean,
SQL Server/Express,
SQL Server DMMS

2. Cài đặt trên Mac/Linux

Tải và cài đặt:

Netbean,
Docker,
SQL Server for mac,
Azure Data Studio

3. Fix lỗi khi kết nối trên Windows

Lỗi do localhost TCP/IP cổng 1433:

<https://www.youtube.com/watch?v=tFXZiHMZJmg>

Lỗi do login fail 'sa':

<https://www.youtube.com/watch?v=ftVcBoZRAMA>

Cấu hình SQL Server cho
phép kết nối trong Java JDBC:

https://www.youtube.com/watch?v=lgD4VvS8_b0

CHƯƠNG 1. LẬP TRÌNH JDBC

MỤC TIÊU

- ⊙ Kết thúc chương này, sinh viên có khả năng:
 - ❖ Trình bày được các nội dung lý thuyết cơ bản về JDBC
 - ❖ Phân biệt được các trình điều khiển JDBC: Type 1, Type 2, Type 3, Type 4
 - ❖ Các bước kết nối với JDBC
 - ❖ Các thao tác cơ bản của JDBC
 - Statement
 - PreparedStatement
 - CallableStatement
 - ResultSet
 - Database Metadata
 - ❖ Quản lý transaction

Giới thiệu về JDBC

- JDBC là chuẩn kết nối CSDL, cung cấp các interface & class nhằm tạo cơ sở cho các ứng dụng Java tương tác với các hệ quản trị CSDL
- Tập hợp các lớp thực thi theo chuẩn JDBC để tương tác với 1 CSDL, cụ thể gọi là JDBC driver
- Phần lớn ý tưởng của JDBC kế thừa từ chuẩn kết nối ODBC của Microsoft



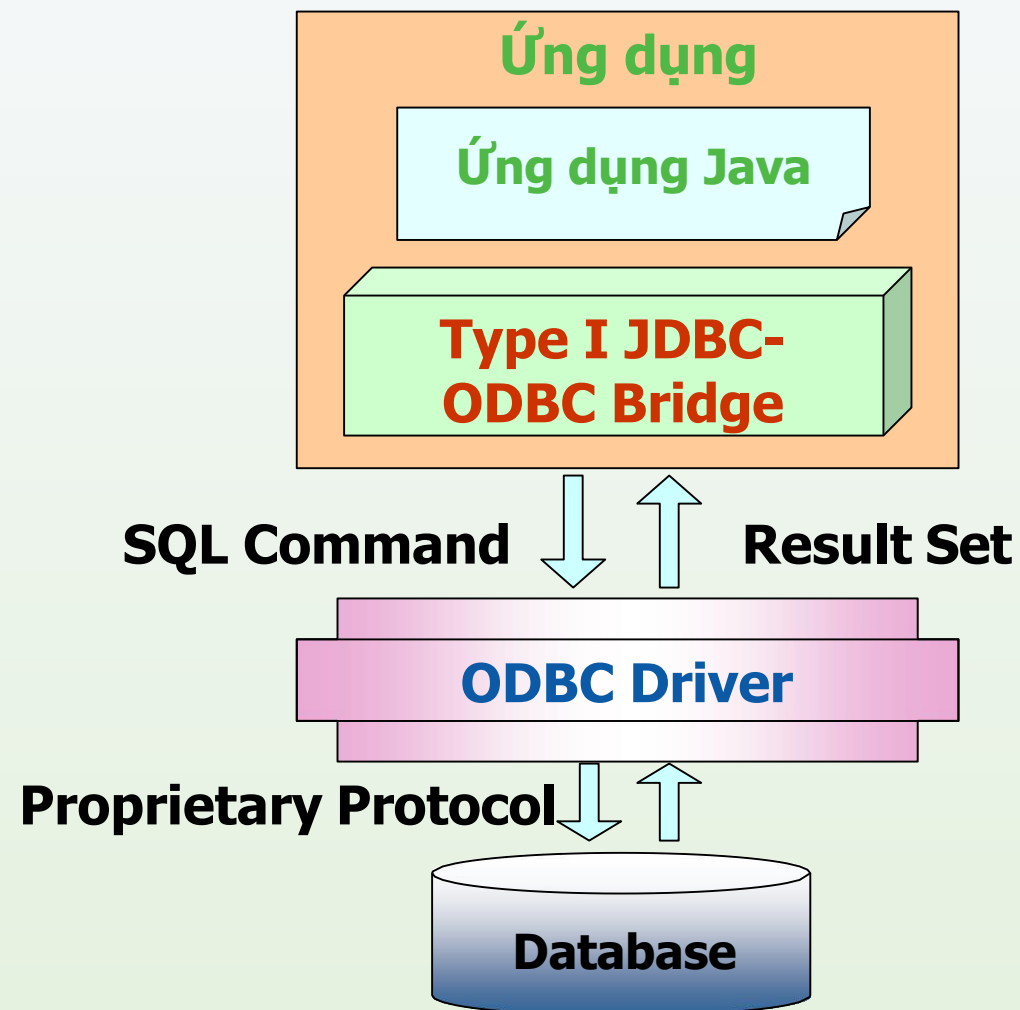
Các loại JDBC

TYPE 1 JDBC/ ODBC

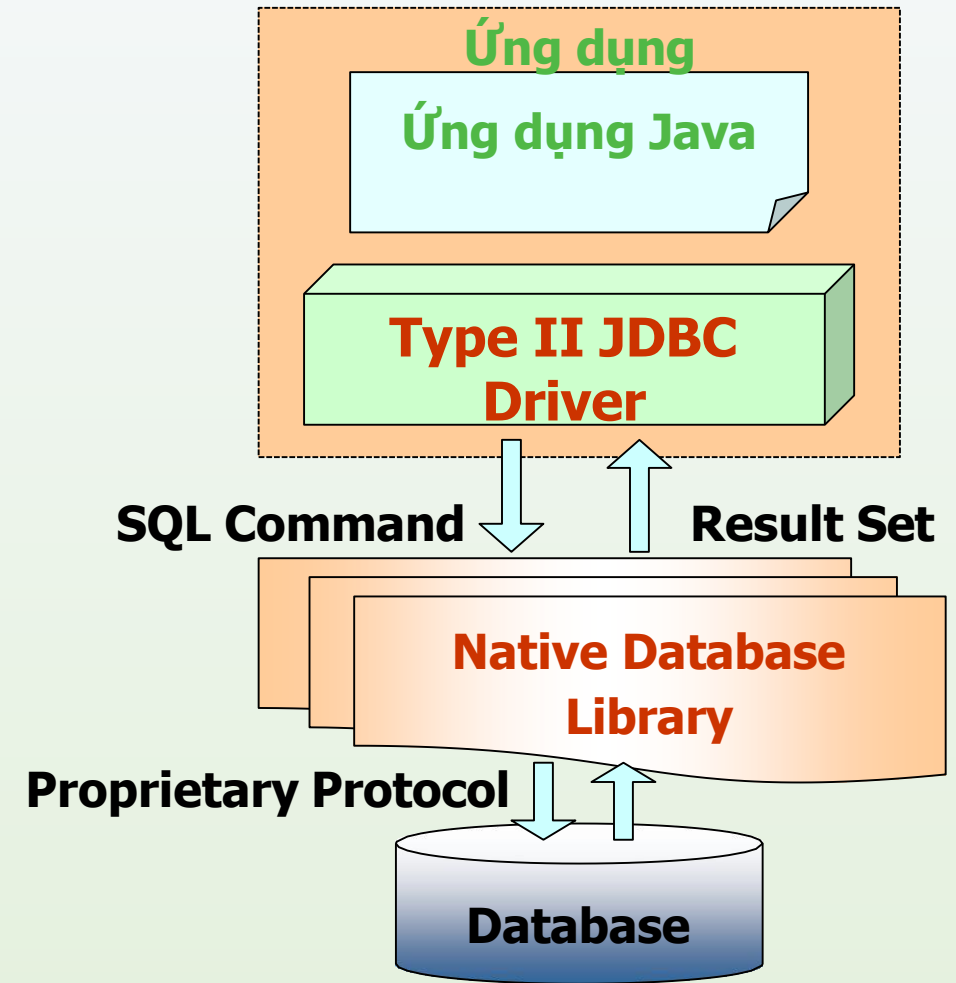
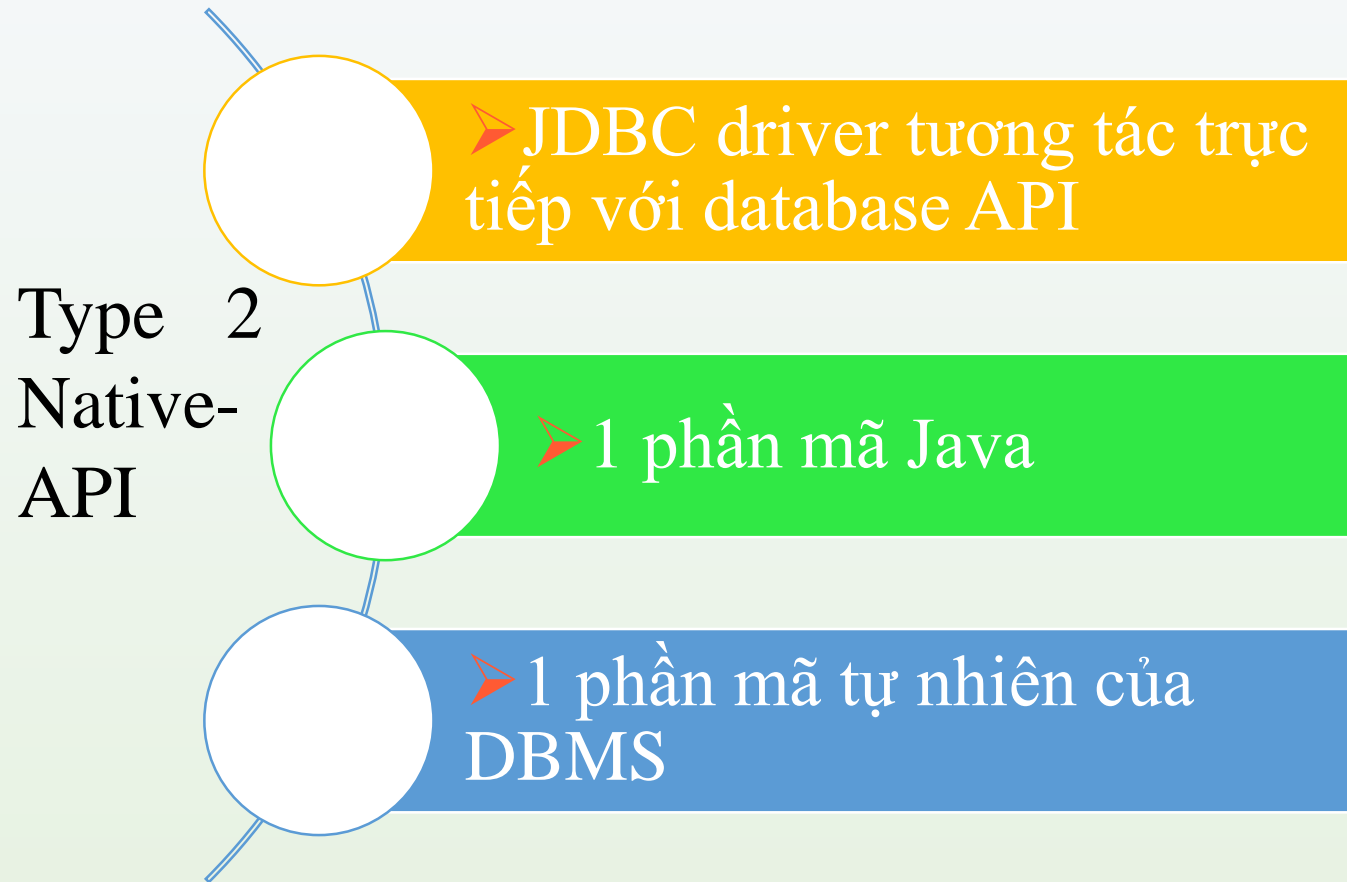
➤ Được cung cấp miễn phí bởi Sun-jdk

➤ Có thể truy xuất bất kỳ DBMS nào được hỗ trợ bởi ODBC driver

➤ Tính khả chuyển cao nhưng kém hiệu quả



Các loại JDBC



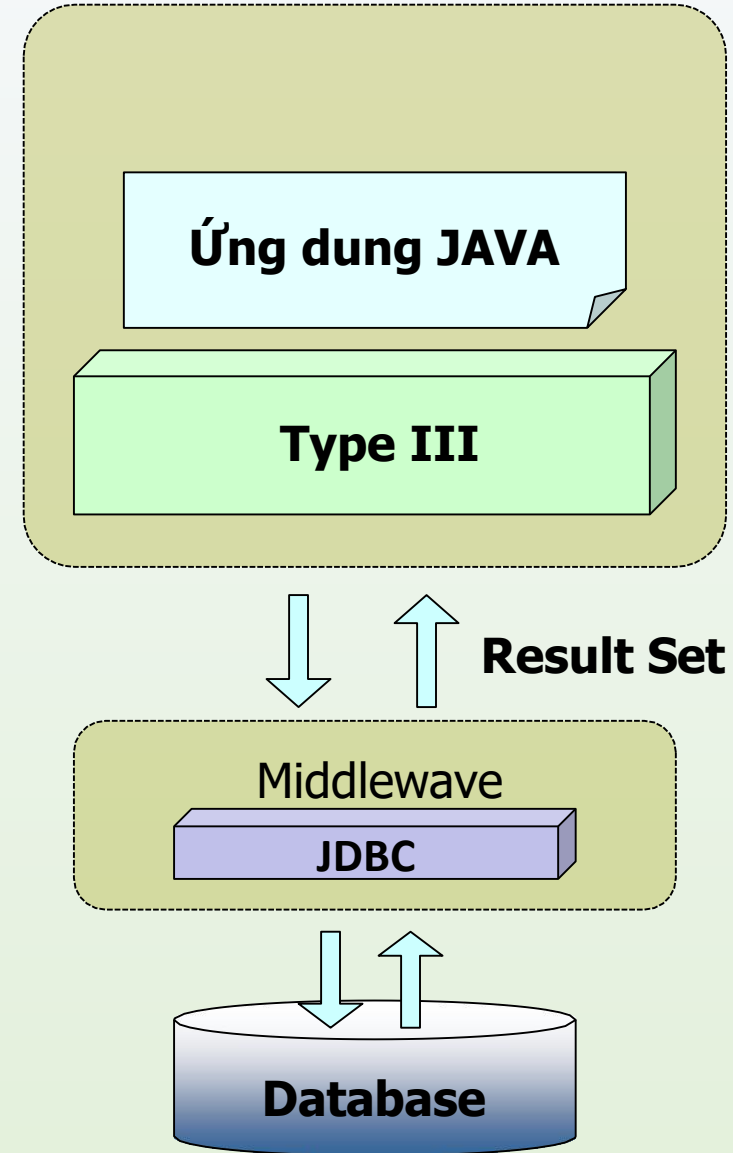
Các loại JDBC

Type 3:
Open
Protocol
-Net

➤ Tương tác với nhiều DBMS theo giao thức mở

➤ 100% Java code

➤ Cài đặt driver cả 2 phía client & server



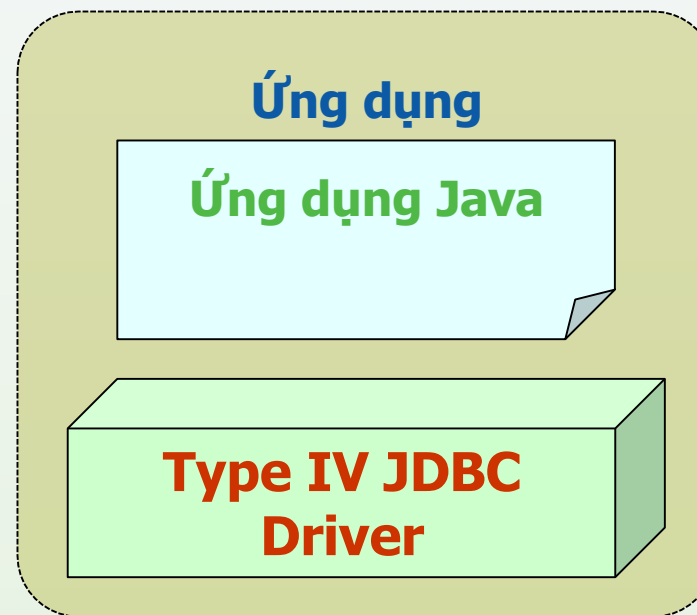
Các loại JDBC

Type 4:
Proprietary-
Protocol
Net

➤ 100% java

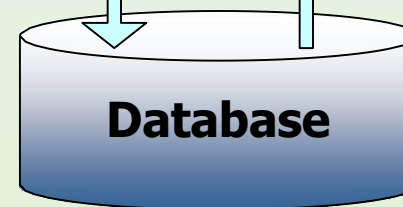
➤ Truy xuất trực tiếp DBMS
theo giao thức độc quyền

➤ Hiệu quả nhất

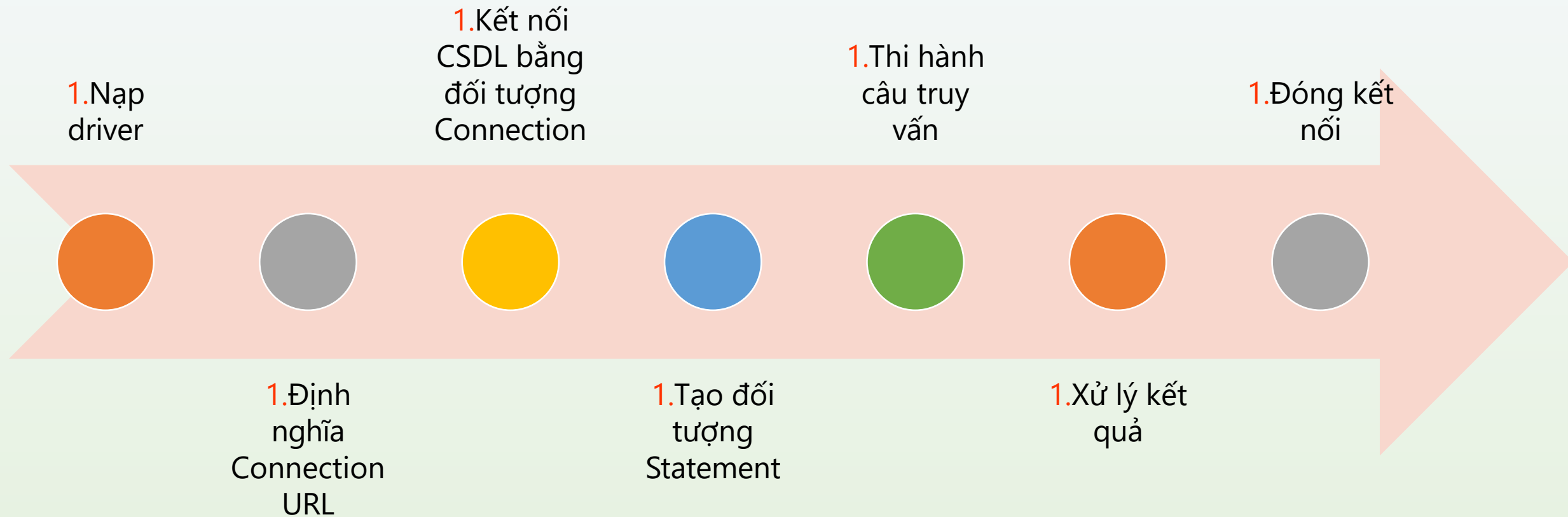


Các câu lệnh SQL, dùng
Proprietary protocol

Result Set, dùng Proprietary
protocol



7 bước kết nối với JDBC



Ví dụ về 7 bước kết nối với JDBC:

Bước 1. Nạp driver

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

Bước 2. Định nghĩa Connection URL

```
// Bai1 la database trong SQL Server  
String url = "jdbc:sqlserver://localhost:1433;databaseName=Bai1";
```

(mỗi loại driver cho 1 loại CSDL sẽ có thay đổi)

Bước 3. Kết nối CSDL bằng đối tượng Connection

```
String user = "sa";  
String password = "Abcde@12345"; // pass trong kích hoạt connection  
Connection conn = DriverManager.getConnection(url, user, password);
```

Ví dụ về 7 bước kết nối với JDBC:

Bước 4. Tạo đối tượng Statement

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

Bước 5. Thi hành câu truy vấn

```
String sql1 = "select * from LOP";  
ResultSet rs =sttm.executeQuery(sql1);  
  
// Để cập nhật, sửa đổi (modify) sử dụng phương thức  
// executeUpdate (cho các lệnh UPDATE, INSERT,DELETE)  
String sql2 = "insert LOP values('L04', 'JAVA')";  
int rowEffect = sttm.executeUpdate(sql2);
```

// Để tạo 1 table, xóa 1 table... ta sử dụng phương thức `execute`

```
String sql3 = "drop table LOP ";
```

```
sttm.execute(sql3);
```

Ví dụ về 7 bước kết nối với JDBC:

Bước 6. Xử lý kết quả

```
while(rs.next()) { // có thể đọc được  
    System.out.println(rs.getString(1) + rs.getInt(2)) ;  
}
```

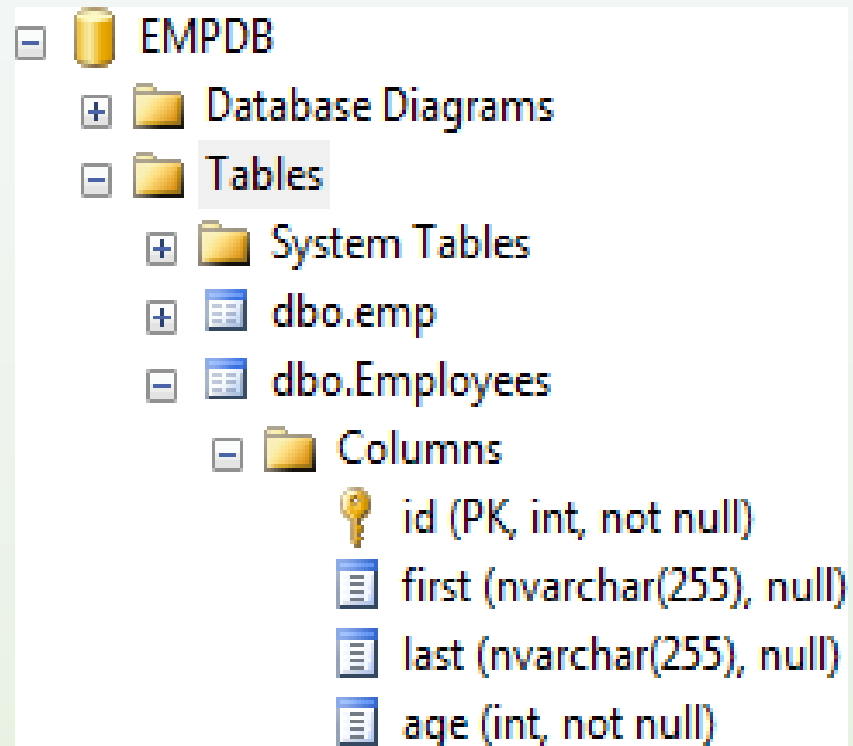
- Cột đầu tiên đánh số là 1
- Có thể dùng tên cột: `rs.getString("TenLop");`
- ResultSet cung cấp 1 số phương thức :
`getString(int)` , `getInt(int)` , `getLong(int)` , `getObject(int)` ,
`getDate(int)`

Bước 7. Đóng kết nối

```
sttm.close();  
conn.close();
```

Ví dụ 1:

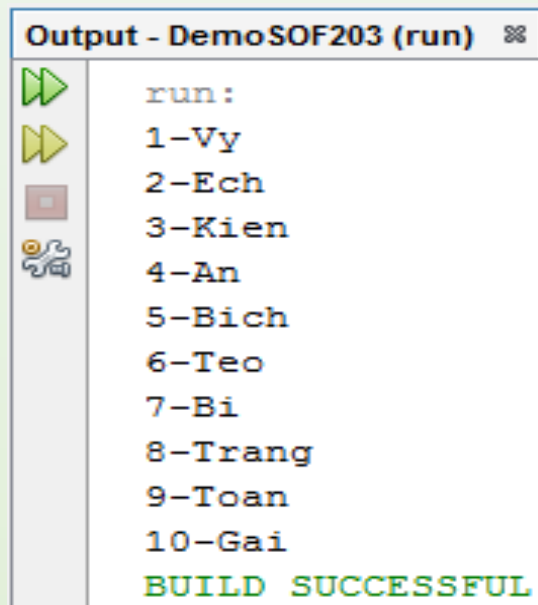
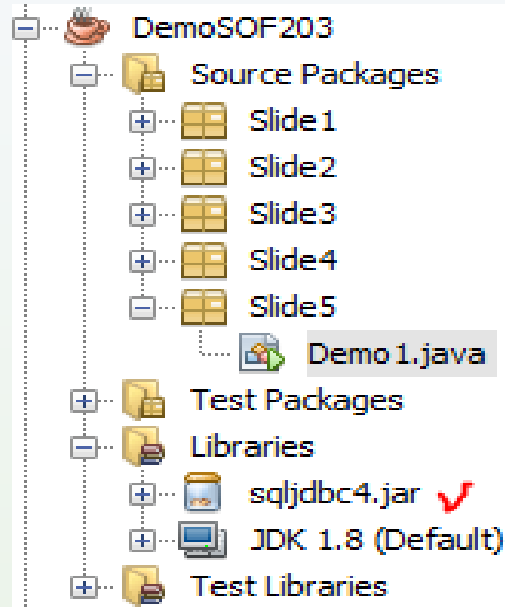
Database



| |
|-----------------------------|
| EMPDB |
| Database Diagrams |
| Tables |
| System Tables |
| dbo.emp |
| dbo.employees |
| Columns |
| id (PK, int, not null) |
| first (nvarchar(255), null) |
| last (nvarchar(255), null) |
| age (int, not null) |

| SCD050718.EMPDB - dbo.employees | | | | |
|---------------------------------|------|-------|------------------|------|
| | id | first | last | age |
| | 1 | Vy | Nguyen Thi Tuong | 35 |
| | 2 | Ech | Tu | 36 |
| | 3 | Kien | Le Phung Hieu | 6 |
| | 4 | An | Le Phung Hieu | 3 |
| | 5 | Bich | Le Thi Ngoc | 20 |
| | 6 | Teo | Le Van | 20 |
| | 7 | Bi | Nguyen Van | 20 |
| | 8 | Trang | Le Thi | 25 |
| | 9 | Toan | Nguyen Van | 20 |
| | 10 | Gai | Nguyen Thi | 21 |
| ►* | NULL | NULL | NULL | NULL |

Ví dụ 1:



```
public class Demo1 {
    public static void main(String[] args) {
        try{
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            String url="jdbc:sqlserver://localhost:1433;databaseName=EMPDB";
            Connection con = DriverManager.getConnection(url,"sa","");
            Statement stm = con.createStatement();
            String sql="select * from Employees";
            ResultSet rs = stm.executeQuery(sql);
            while(rs.next()) {
                System.out.println(rs.getInt(1)+"-"+ rs.getString("first"));
            }
            rs.close();stm.close();con.close();
        }catch(Exception e){
            System.out.println(e);
        }
    }
}
```

Lưu ý: thêm sqljdbc4.jar vào Libraries như hình

DEMO 1

CÂU LỆNH STATEMENT

Ba 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

Sử dụng kết nối connection để tạo câu lệnh

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

```
PreparedStatement pstmt = conn.prepareStatement(sqlString);
```

```
CallableStatement cstmt = con.prepareCall(sqlString);
```

Câu lệnh Statement có thể được sử dụng nhiều lần cho những tác vụ khác nhau, những câu lệnh SQL không liên quan nhau

THI HÀNH STATEMENT

Có 3 cách thi hành Statement

➤ executeQuery()

Dùng để thi hành các câu lệnh truy vấn Select...from...where

+ Trả về kết quả truy vấn qua đối tượng ResultSet

+ `ResultSet rs = sttm.executeQuery("SELECT * FROM Employee");`

```
public class Statement1 {  
    public static void main(String[] args) {  
        try{  
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");  
            String url="jdbc:sqlserver://localhost:1433;databaseName=EMPDB";  
            Connection con = DriverManager.getConnection(url,"sa","");  
            Statement stm = con.createStatement();  
            String sql="SELECT id, first, last, age FROM Employees";  
            ResultSet rs = stm.executeQuery(sql);  
            while(rs.next()){  
                int id = rs.getInt("id");  
                int age = rs.getInt("age");  
                String first = rs.getString("first");  
                String last = rs.getString("last");  
  
                System.out.print("ID: " + id);  
                System.out.print(", Age: " + age);  
                System.out.print(", First: " + first);  
                System.out.println(", Last: " + last);  
            }  
            rs.close();stm.close();con.close();  
        }catch(Exception e){  
            System.out.println(e);  
        }  
    }  
}
```

Output - DemoSOF203 (run)

```
run:  
ID: 1, Age: 35, First: Vy, Last: Nguyen Thi Tuong  
ID: 2, Age: 36, First: Ech, Last: Tu  
ID: 3, Age: 6, First: Kien, Last: Le Phung Hieu  
ID: 4, Age: 3, First: An, Last: Le Phung Hieu  
ID: 5, Age: 20, First: Bich, Last: Le Thi Ngoc  
ID: 6, Age: 20, First: Teo, Last: Le Van  
ID: 7, Age: 20, First: Bi, Last: Nguyen Van  
ID: 8, Age: 25, First: Trang, Last: Le Thi  
ID: 9, Age: 20, First: Toan, Last: Nguyen Van  
ID: 10, Age: 21, First: Gai, Last: Nguyen Thi  
BUILD SUCCESSFUL (total time: 3 seconds)
```

THI HÀNH STATEMENT

Có 3 cách thi hành Statement

➤ executeQuery()

```
public class Statement1 {  
    public static void main(String[] args) {  
        try{  
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");  
            String url="jdbc:sqlserver://localhost:1433;databaseName=EMPDB";  
            Connection con = DriverManager.getConnection(url,"sa","");  
            Statement stm = con.createStatement();  
            String sql="SELECT id, first, last, age FROM Employees";  
            ResultSet rs = stm.executeQuery(sql);  
            while(rs.next()){  
                int id = rs.getInt("id");  
                int age = rs.getInt("age");  
                String first = rs.getString("first");  
                String last = rs.getString("last");  
  
                System.out.print("ID: " + id);  
                System.out.print(", Age: " + age);  
                System.out.print(", First: " + first);  
                System.out.println(", Last: " + last);  
            }  
            rs.close();stm.close();con.close();  
        }catch(Exception e){  
            System.out.println(e);  
        }  
    }  
}
```

Output - DemoSOF203 (run) %

```
run:  
ID: 1, Age: 35, First: Vy, Last: Nguyen Thi Tuong  
ID: 2, Age: 36, First: Ech, Last: Tu  
ID: 3, Age: 6, First: Kien, Last: Le Phung Hieu  
ID: 4, Age: 3, First: An, Last: Le Phung Hieu  
ID: 5, Age: 20, First: Bich, Last: Le Thi Ngoc  
ID: 6, Age: 20, First: Teo, Last: Le Van  
ID: 7, Age: 20, First: Bi, Last: Nguyen Van  
ID: 8, Age: 25, First: Trang, Last: Le Thi  
ID: 9, Age: 20, First: Toan, Last: Nguyen Van  
ID: 10, Age: 21, First: Gai, Last: Nguyen Thi  
BUILD SUCCESSFUL (total time: 3 seconds)
```

THI HÀNH STATEMENT

Có 3 cách thi hành Statement

➤ executeUpdate()

Dùng cho câu lệnh cập nhật dữ liệu

+ Trả về số bản ghi chịu ảnh hưởng bởi câu lệnh UPDATE, INSERT, DELETE

+ Trả về 0, có nghĩa là:

Không có bản ghi nào bị ảnh hưởng

Thực thi câu lệnh DDL định nghĩa dữ liệu

Ví dụ:

THI HÀNH STATEMENT

Có 3 cách thi hành Statement

➤ executeUpdate()

```
public class Statement2 {  
    public static void main(String[] args) {  
        try{  
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");  
            String url="jdbc:sqlserver://localhost:1433;databaseName=EMPDB";  
            Connection con = DriverManager.getConnection(url,"sa","");  
            Statement stm = con.createStatement();  
            //String sql="INSERT INTO Employees VALUES (11, N'Ti', N'Phan Văn', 22);";  
            String sql="delete from Employees where id=11";  
            int rows = stm.executeUpdate(sql);  
            System.out.println("Rows: "+rows);  
            stm.close();con.close();  
        }catch(Exception e){  
            System.out.println(e);  
        }  
    }  
}
```

Output - Demo\$OF203 (run) ☒



run:



Rows: 1



BUILD SUCCESSFUL (total time: 1 second)

THI HÀNH STATEMENT

Có 3 cách thi hành Statement

➤ execute()

- ✓ Khi không biết rõ câu lệnh là truy vấn hay cập nhật
- ✓ Dùng cho các trường hợp thực thi SQL động
- ✓ Trả về true nếu câu lệnh là truy vấn
 - Gọi getResultSet() để nhận được kết quả truy vấn
 - Gọi getUpdatedCount() để biết số bản ghi đã cập nhật

Sử dụng các loại statement ở trên, với CSDL đã xây dựng được ở Demo1, hãy thực hiện các tính năng:

- Hiển thị thông tin các nhân viên
- Cập nhật tuổi cho nhân viên
- Xoá nhân viên
- Thêm nhân viên mới

PREPAREDSTATEMENTS

- Sử dụng PreparedStatement để tăng hiệu quả thi hành câu lệnh SQL
- Câu lệnh SQL sẽ được biên dịch 1 lần trước khi được gọi thi hành nhiều lần
- Thay đổi đối số mỗi lần thi hành

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

Số 1: chỉ định cho dấu ? thứ nhất ở câu lệnh sql, "Danang" là giá trị cho Address ở câu lệnh sql bên trên khi thực thi

- Sau khi thiết lập giá trị đối số, chúng được giữ nguyên cho đến khi thiết lập giá trị mới hoặc gọi phương thức clearParameters() để xóa giá trị các đối số

Số 2: chỉ định cho dấu ? thứ hai ở câu lệnh sql, 1001 là giá trị cho CustNo ở câu lệnh sql bên trên khi thực thi

PREPAREDSTATEMENTS

```
public class DemoPreparedStatement {  
    public static void main(String[] args) {  
        try{  
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");  
            String url="jdbc:sqlserver://localhost:1433;databaseName=EMPDB";  
            Connection con = DriverManager.getConnection(url,"sa","");  
            String sql="UPDATE Employees set age=? WHERE id=?";  
            PreparedStatement stm = con.prepareStatement(sql);  
            stm.setInt(1, 40);  
            stm.setInt(2, 7);  
            int rows = stm.executeUpdate();  
            System.out.println("Rows: " + rows);  
            stm.close();con.close();  
        }catch(Exception e){  
            System.out.println(e);  
        }  
    }  
}
```

Output - DemoSOF203 (run) ☒



run:



Rows: 1



BUILD SUCCESSFUL (total time: 4 seconds)

Sử dụng CSDL đã xây dựng được ở Demo2, hãy chỉnh sửa lại tính năng:

- Cập nhật lại tuổi age của nhân viên theo mã nhập từ bàn phím
- Xoá nhân viên có mã được nhập từ bàn phím

CALLABLE STATEMENT

- CallableStatement cung cấp câu lệnh gọi thi hành các thủ tục đã cài đặt sẵn trên DBMS
- Cú pháp:
 - { Call procedure_name(arg1,arg2,...) }
 - { ? = call procedure_name arg1, arg2,... }

Dấu ? Thay chỗ cho các đối số

Các đối số có thể là input(IN parameters), output(OUT parameters), hoặc cả 2 (INOUT parameters)

CALLABLE STATEMENT

```
CallableStatement cstmt = conn.prepareCall("{call Proc(?, ?)}");
```

Truyền đối số IN bằng hàm setxxx() kế thừa từ PreparedStatement
Đăng ký đối số OUT trước khi thi hành thủ tục

```
registerOutParameter(1, Types.VARCHAR);
```

Đối số INOUT

```
Stmt1.setString(1, "00000");
```

```
Stmt1.registerOutParameter(1, Types.VARCHAR);
```

Các stored procedure không phù hợp trong môi trường phân tán phức hợp vì nó gắn chặt với 1 DBMS cụ thể

CALLABLE STATEMENT

```
public class DemoPrepareCall {  
    public static void main(String[] args) {  
        try{  
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");  
            String url="jdbc:sqlserver://localhost:1433;databaseName=EMPDB";  
            Connection con = DriverManager.getConnection(url,"sa","");  
            String sql="{call getEmpName (?, ?)}";  
            CallableStatement stmt = con.prepareCall(sql);  
            int empID = 3;  
            stmt.setInt(1, empID);  
            stmt.registerOutParameter(2, java.sql.Types.NVARCHAR);  
            System.out.println("Executing stored procedure...");  
            stmt.execute();  
            String empName = stmt.getString(2);  
            System.out.println("Emp name with ID (" + empID + ") is " + empName);  
            stmt.close();con.close();  
        }catch(Exception e){  
            System.out.println(e);  
        }  
    }  
}
```

| id | first | last | age |
|----|-------|------------------|-----|
| 1 | Vy | Nguyen Thi Tuong | 35 |
| 2 | Ech | Tu | 36 |
| 3 | Kien | Le Phung Hieu | 6 |
| 4 | An | Le Phung Hieu | 3 |
| 5 | Bich | Le Thi Ngoc | 20 |
| 6 | Teo | Le Van | 20 |
| 7 | Bi | Nguyen Van | 40 |
| 8 | Trang | Le Thi | 25 |
| 9 | Toan | Nguyen Van | 20 |
| 10 | Gai | Nguyen Thi | 21 |


```
CREATE PROCEDURE getEmpName  
    @EMP_ID INT, @EMP_FIRST NVARCHAR(255) OUTPUT  
AS  
BEGIN  
    SELECT @EMP_FIRST = first  
    FROM Employees  
    WHERE ID = @EMP_ID  
END
```

Output - DemoSOF203 (run) ☒

```
run:  
Executing stored procedure...  
Emp name with ID (3) is Kien  
BUILD SUCCESSFUL (total time: 2 seconds)
```

RESULTSET

- ResultSet cho phép truy xuất đến dữ liệu trả về từ kết quả truy vấn database
- Truy xuất lần lượt từng trường của bản ghi bằng 1 con trỏ chỉ đến vị trí hiện hành trong ResultSet
- Gọi hàm next() để di chuyển con trỏ hiện hành đến hàng kế tiếp của ResultSet
next() trả về true nghĩa là còn dữ liệu để đọc
- Sử dụng cấu trúc lặp sau đây để duyệt 1 ResultSet

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

RESULTSET

- Dữ liệu tại mỗi trường của bản ghi được đọc bởi hàm get() theo mẫu

Type getType(int String)

- Đối số là thứ tự cột – bắt đầu từ 1 hoặc tên cột
- Kiểu của type có thể là int, double, String, Date, ... tùy ý

```
String isbn = rs.getString(1); //Column 1  
double price = rs.getDouble("Price");
```

- Lưu ý
 - ❖ ResultSet gắn liền với Connection đến CSDL
 - ❖ Forward only theo mặc định, tức là di chuyển tới chứ không lui được

RESULTSET

```
public class Statement1 {  
    public static void main(String[] args) {  
        try{  
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");  
            String url="jdbc:sqlserver://localhost:1433;databaseName=EMPDB";  
            Connection con = DriverManager.getConnection(url,"sa","");  
            Statement stm = con.createStatement();  
            String sql="SELECT id, first, last, age FROM Employees";  
            ResultSet rs = stm.executeQuery(sql);  
            while(rs.next()){  
                int id = rs.getInt("id");  
                int age = rs.getInt("age");  
                String first = rs.getString("first");  
                String last = rs.getString("last");  
  
                System.out.print("ID: " + id);  
                System.out.print(", Age: " + age);  
                System.out.print(", First: " + first);  
                System.out.println(", Last: " + last);  
            }  
            rs.close();stm.close();con.close();  
        }catch(Exception e){  
            System.out.println(e);  
        }  
    }  
}
```

Output - DemoSOF203 (run) ✖



run:

```
ID: 1, Age: 35, First: Vy, Last: Nguyen Thi Tuong  
ID: 2, Age: 36, First: Ech, Last: Tu  
ID: 3, Age: 6, First: Kien, Last: Le Phung Hieu  
ID: 4, Age: 3, First: An, Last: Le Phung Hieu  
ID: 5, Age: 20, First: Bich, Last: Le Thi Ngoc  
ID: 6, Age: 20, First: Teo, Last: Le Van  
ID: 7, Age: 20, First: Bi, Last: Nguyen Van  
ID: 8, Age: 25, First: Trang, Last: Le Thi  
ID: 9, Age: 20, First: Toan, Last: Nguyen Van  
ID: 10, Age: 21, First: Gai, Last: Nguyen Thi  
BUILD SUCCESSFUL (total time: 3 seconds)
```


RESULTSET & DATABASE METADATA

- ResultSetMetadata là lớp cung cấp thông tin về bản thân ResultSet

```
ResultSet rs = stmt.executeQuery(sqlString);  
ResultSetMetadata rsmd = rs.getMetaData();  
Int numberOfColumns = rsmd.getColumnCount();  
String colName = getColumnName(int column);
```

- DatabaseMetadata là các lớp cung cấp thông tin về bản thân CSDL
 - ✓ Số table
 - ✓ Cấu trúc các table
 - ✓ Các phiên bản thực thi JDBC driver của các hãng không giống nhau

RESULTSET & DATABASE METADATA

```
public class DemoDatabaseMetaData {  
    public static void main(String[] args) throws ClassNotFoundException {  
        try {  
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");  
            String dbURL = "jdbc:sqlserver://localhost;databaseName=EMPDB;user=sa;password=";  
            Connection conn = DriverManager.getConnection(dbURL);  
            if (conn != null) {  
                System.out.println("Connected");  
                DatabaseMetaData dm = (DatabaseMetaData) conn.getMetaData();  
                System.out.println("Driver name: " + dm.getDriverName());  
                System.out.println("Driver version: " + dm.getDriverVersion());  
                System.out.println("Product name: " + dm.getDatabaseProductName());  
                System.out.println("Product version: " + dm.getDatabaseProductVersion());  
            }  
        } catch (SQLException ex) {  
            System.err.println("Cannot connect to the database");  
        }  
    }  
}
```

Output - DemoSOF203 (run) ✖



run:



Connected



Driver name: Microsoft SQL Server 2005 JDBC Driver



Driver version: 2.0.1008.2

Product name: Microsoft SQL Server

Product version: 10.00.1600

BUILD SUCCESSFUL (total time: 2 seconds)

QUẢN LÝ TRANSACTION

- Tắt Autocommit mode
- Theo mặc định, JDBC thực thi trọn vẹn (commit) các câu lệnh SQL một khi nó được chuyển đến database, gọi là autocommit
- Một số ứng dụng mang đặc điểm transaction-yêu cầu các tác vụ thi hành hoặc cả gói hoặc không gì cả
 - ✓ Tắt chế độ autocommit để thực hiện quản lý transaction theo đặc điểm của ứng dụng
 - ✓ Lớp Connection cung cấp hàm setAutoCommit() để bật tắt chế độ autocommit
 - ✓ Câu lệnh SQL đầu tiên đồng thời bắt đầu 1 transaction, kết thúc bằng commit() hoặc rollback()

```
conn.setAutoCommit(false);
```

```
s = conn.createStatement();
```

```
s.executeUpdate(sqlString)
```

```
conn.commit(); // hoặc rollback();
```

Demo tính năng commit và rollback

Bài tập

Sinh viên thực hiện các bài tập trong sách