

Truy vấn MySQL qua Java với JDBC

Giảng viên :Cao Le Thanh

Mục tiêu bài học



Nội dung

- ❖ Giới thiệu chung về JDBC
- ❖ Trình điều khiển JDBC
 - ❖ Phân loại
 - ❖ Database URL
- ❖ Các lớp tác vụ cơ bản của JDBC
 - ❖ Statement
 - ❖ Res
- ❖ Quản lý transaction
- ❖ Xử lý đa người dùng



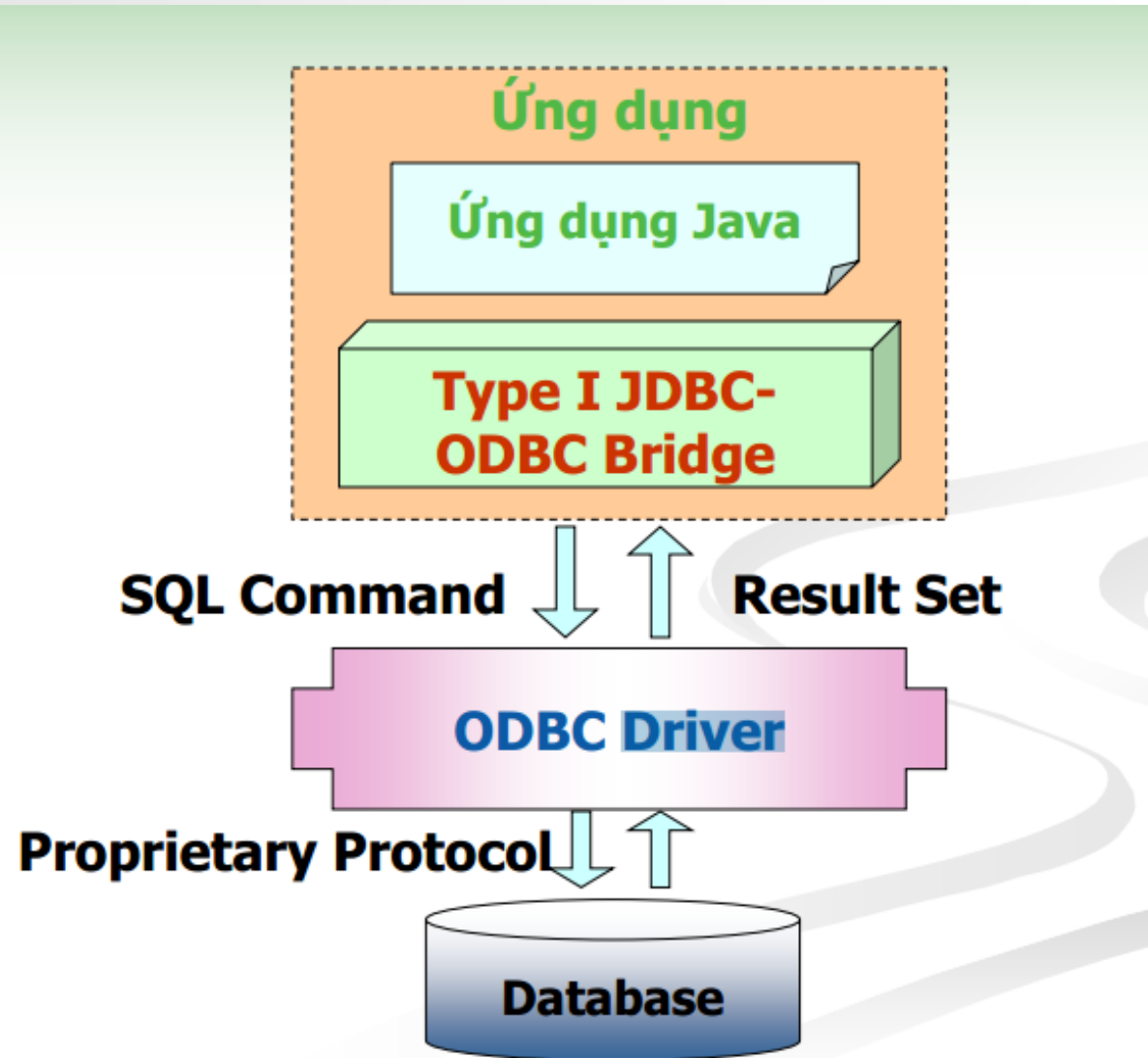
GIỚI THIỆU CHUNG VỀ JDBC

- ❖ 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

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

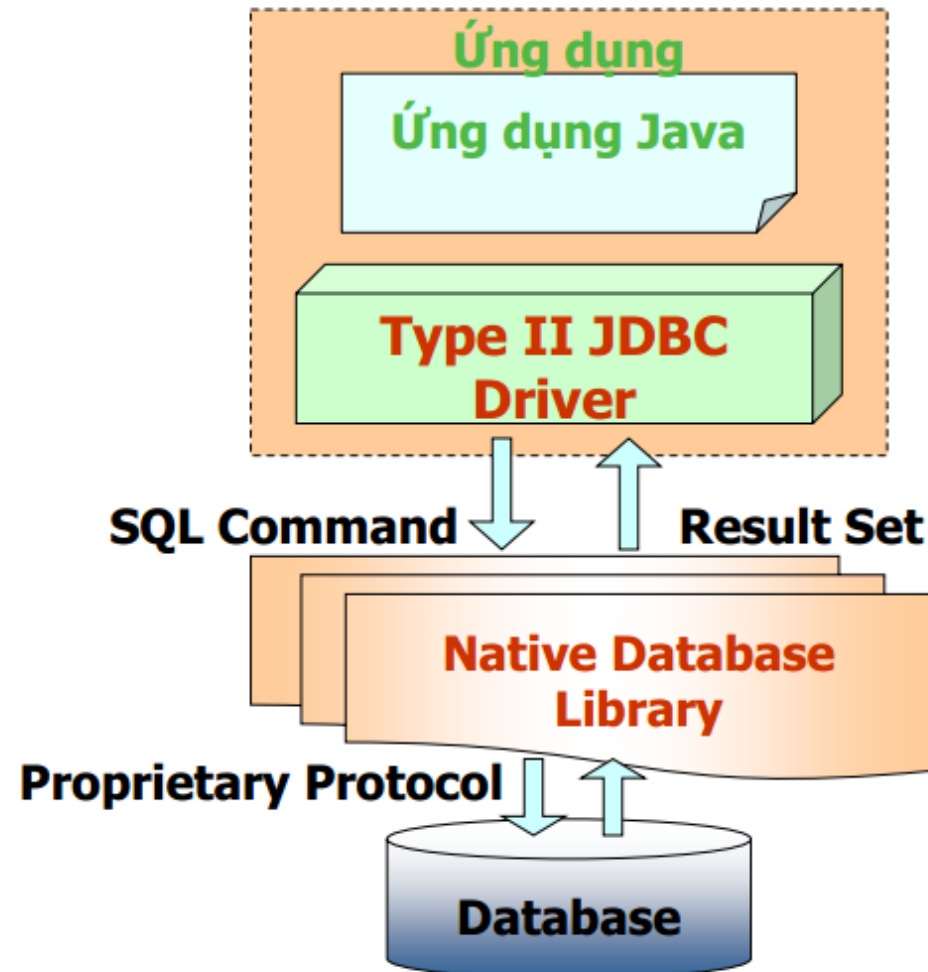
JBDC – ODBC Bridge, plus ODBC driver



Type 2 Native - API

- ❖ JDBC driver tương tác trực tiếp với database API
 - 1 phần mã Java
 - 1 phần mã tự nhiên của DBMS

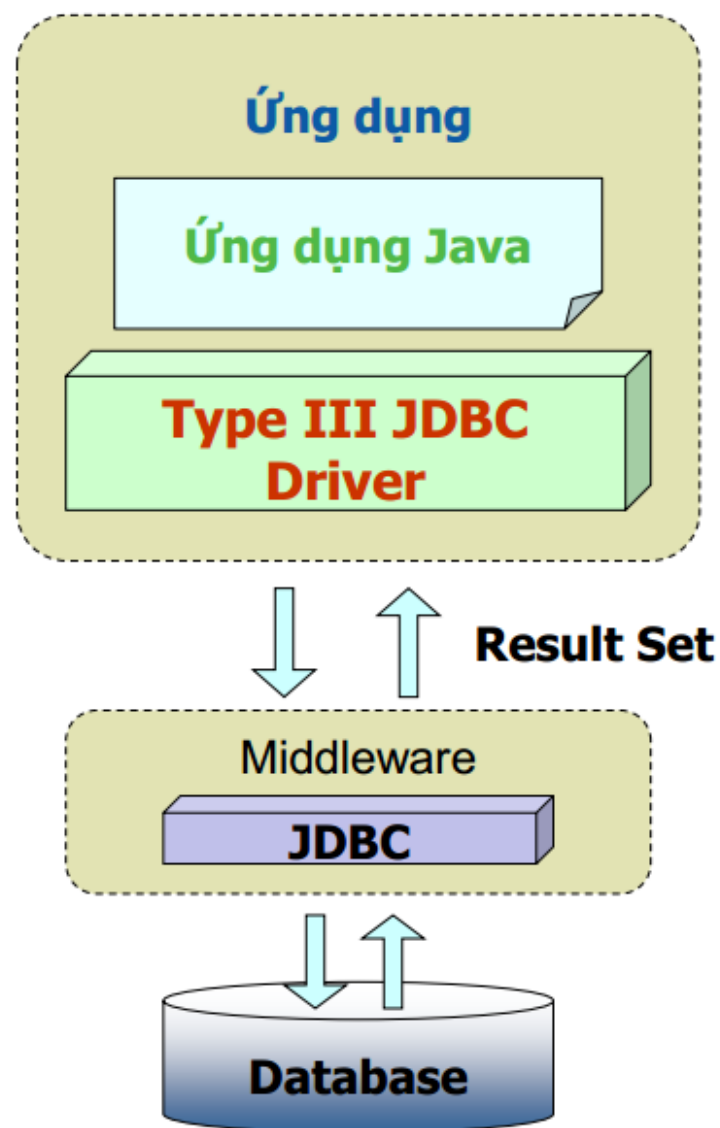
Native-API , partly Java driver



Type 3: Open Protocol - Net

- ❖ Tương tác với nhiều DBMS theo giao ước mở
 - 100% Java code
 - Cài đặt driver cả 2 phía client & server

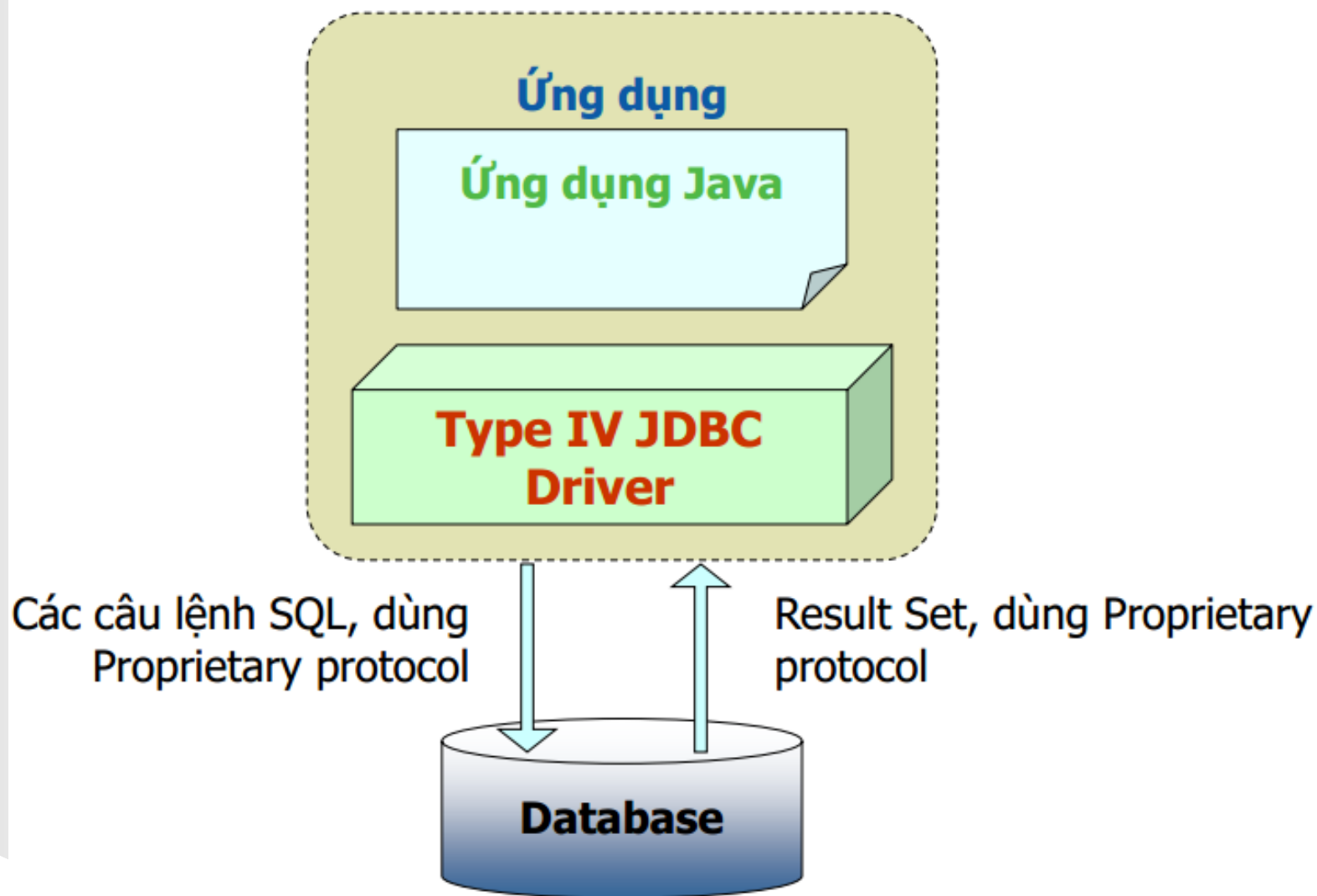
JBDC – net , pure Java driver



Type 4 : Proprietary – Protocol Net

- ❖ 100% java
- ❖ Truy xuất trực tiếp DBMS theo giao thức độc quyền
- ❖ Hiệu quả nhất

Native protocol – pure Java driver



7 bước kết nối với JBDC

- ❖ Nạp driver
- ❖ Định nghĩa Connection URL
- ❖ Kết nối CSDL bằng đối tượng Connection
- ❖ Tạo đối tượng Statement
- ❖ Thi hành câu truy vấn
- ❖ Xử lý kết quả
- ❖ Đóng kết nối

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 = con.createStatement(String);`
- `PreparedStatement ps = con.prepareStatement(String);`
- `CallableStatement cs = con.prepareCall(String);`

❖ 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()`
- `executeUpdate()`
- `execute()`

❖ `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 = s.executeQuery("SELECT * FROM lms_users")`

❖ 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
 - Không có bản ghi nào chịu ảnh hưởng
 - Thực thi câu lệnh DDL định nghĩa dữ liệu

❖ execute()

- Khi không biết rõ câu lệnh là truy vấn hay cập nhật DBMS
- 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

PreparedStatement

- ❖ 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);
```

❖ 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

Callable Statement

❖ Ví dụ:

```
CallableStatement cstms = con.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

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 kiểu dữ liệu để đọc, ngược lại **norow**
 - Sử dụng cấu trúc lặp sau đây để duyệt 1 ResultSet

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

Xử lý 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

- `Float price = rs.getDouble("price");` // Column name là price

❖ Lưu ý

- ResultSet gắn liền với Connection đến CSDL

- Forward only theo mặc định

- Chuyển đổi kiểu động

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();`
 - `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

Quản lý Transaction

- ❖ Tắt Autocommit code
- ❖ 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 auto-commit
 - Câu lệnh SQL đầu tiên đồng thời bắt đầu 1 transaction , kết thúc câu lệnh commit() hoặc rollback()

Quản lý Transaction

❖ Ví dụ:

```
con.setAutoCommit(false);
```

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

```
s.executeUpdate(SQLString);
```

```
con.commit(); hoặc rollback();
```

Tổng kết

