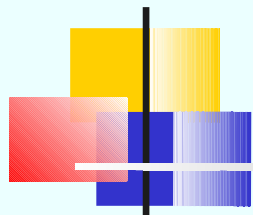




Lập trình mạng

Java kết nối cơ sở dữ liệu

Giảng viên: TS. Nguyễn Mạnh Hùng
Học viện Công nghệ Bưu chính Viễn thông (PTIT)



Nội dung

- Thiết kế CSDL
- Sử dụng lệnh SQL
- Kết nối với DB bằng JDBC
- Dùng lệnh QSL trong Java
- Lấy kết quả ra xử lí
- Làm việc với transaction
- Ví dụ minh họa



Thiết kế CSDL cho ứng dụng



Thiết kế CSDL

- Bước 1: Mỗi đối tượng có thuộc tính cần quản lí thì đề xuất thành một bảng. Thuộc tính của đối tượng thành tên cột của bảng
- Bước 2: Xét quan hệ giữa các bảng:
 - Nếu quan hệ 1-1 thì gộp thành 1 bảng
 - Nếu quan hệ 1-n thì giữ nguyên
 - Nếu quan hệ n-n thì tách thành ít nhất 1 bảng trung gian ở giữa để các bảng chỉ còn quan hệ 1-n
- Bước 3: Thêm khóa ngoài:
 - Giữa hai bảng có quan hệ 1-n, bảng bên n cần thêm một khóa ngoài tham chiếu đến khóa chính của bảng bên 1
- Bước 4: Loại bỏ thuộc tính dư thừa
 - Nếu hai bảng có quan hệ 1-n mà có cùng thuộc tính gây dư thừa, thì loại bỏ thuộc tính bảng bên n
 - Nếu thuộc tính có thể tính toán từ các thuộc tính khác thì loại bỏ



Bài toán

- Một khách sạn (id, tên, địa chỉ, số sao, mô tả) có nhiều phòng (id, tên, hạng phòng, giá niêm yết, mô tả)
- Mỗi phòng có thể được đặt bởi nhiều khách hàng (id, tên, số id, kiểu thẻ id, địa chỉ, mô tả) tại nhiều thời điểm khác nhau
- Mỗi khách hàng có thể đặt nhiều phòng tại nhiều thời điểm khác nhau nhưng chỉ ở 1 phòng tại 1 thời điểm nhất định, xác định 1 giá xác định
- Khách hàng chỉ có thể đặt phòng nếu phòng còn trống trong suốt thời gian khách hàng muốn đặt
- Khi trả phòng, nhân viên in phiếu thanh toán bao gồm tên khách sạn, tên khách hàng, số phòng, hạng phòng, ngày đến, ngày đi và tổng số tiền thanh toán
- Khách hàng có thể thanh toán nhiều lần cho đến trước ngày trả phòng



Thiết kế CSDL (1)

- Trước hết cần có 3 bảng:

tblHotel: id, name, starLevel, address, description

tblRoom: id, name, type, displayPrice, description

tblUser: id, fullName, username, password, idCardNumber, idCardType, address, description

Nhận xét:

- 1 khách sạn có nhiều phòng, 1 phòng chỉ ở trong 1 khách sạn → quan hệ giữa bảng tblHotel và tblRoom là 1-n
 - 1 khách hàng đặt nhiều phòng, 1 phòng cũng có nhiều người đặt → quan hệ giữa tblRoom và tblUser là n-n, → phải chuẩn hóa
- Tạo thêm một bảng đặt chỗ chi tiết **tblBooking**: idRoom, idUser, startDate, endDate, price, description
- quan hệ giữa tblRoom và tblBooking là 1-n (một phòng có nhiều lần đặt), và giữa tblUser và tblBooking cũng là 1-n (một người có nhiều lần đặt)



Thiết kế CSDL (2)

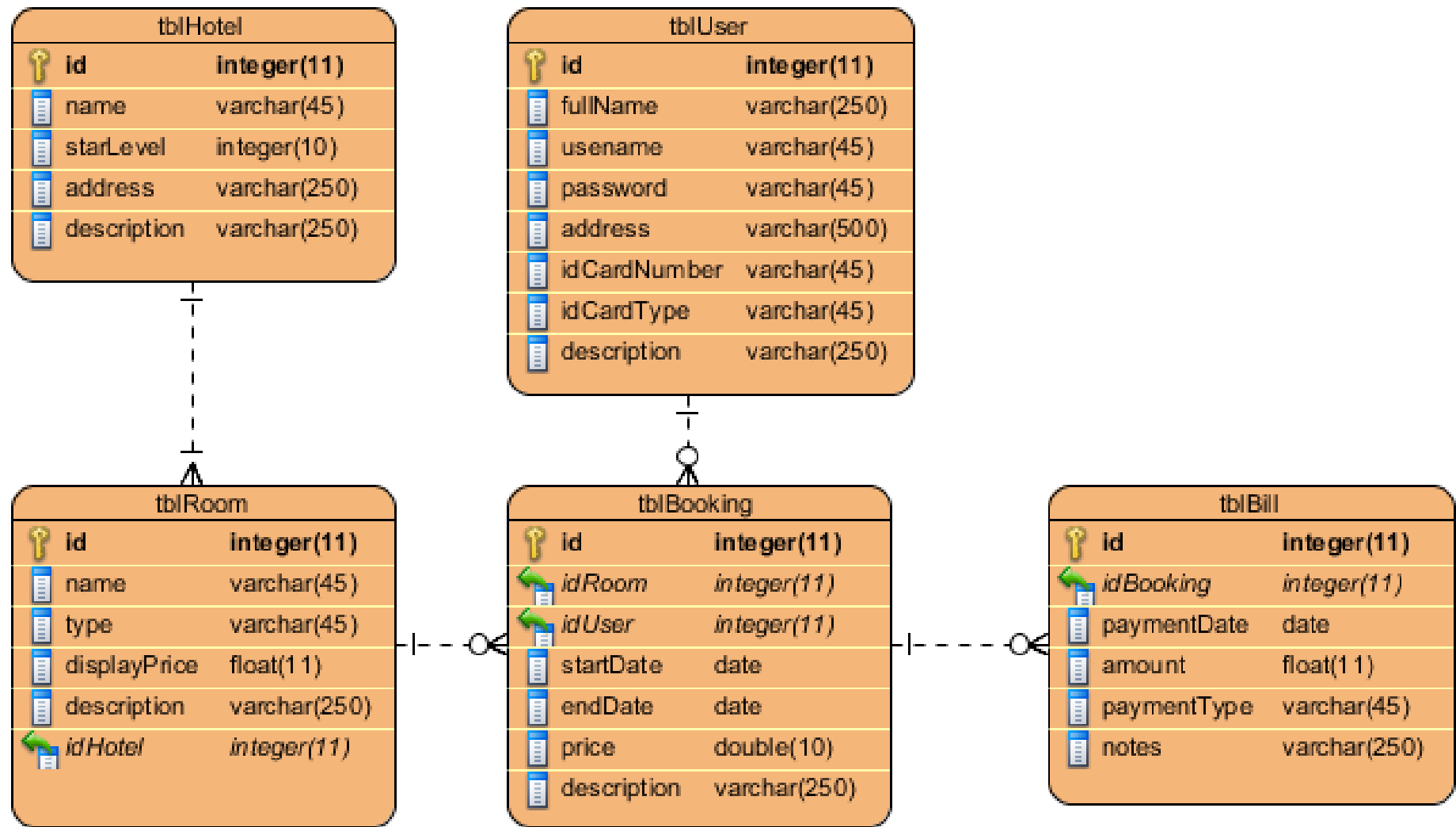
- Thêm 1 bảng:

tblBill: id, idBooking, paymentDate, amount, paymentType, notes

Nhận xét:

- 1 đặt phòng có thể trả tiền nhiều lần (đặt cọc, trả trước, thanh toán khi ở xong) → quan hệ giữa bảng tblBooking và tblBill là 1-n

Thiết kế CSDL (3)





Bài tập (1)

Thiết kế CSDL cho phần mềm quản lí bán vé cho một chuỗi rạp chiếu phim với mô tả như sau:

- Hãng có một chuỗi rạp chiếu phim (Mã rạp, tên rạp, địa chỉ, giới thiệu).
- Mỗi rạp chiếu phim có nhiều phòng chiếu khác nhau (Mã phòng chiếu, đặc điểm phòng chiếu)
- Mỗi phim (Mã phim, tên phim, loại phim, năm sản xuất, mô tả) có thể được chiếu tại nhiều phòng chiếu khác nhau vào nhiều thời điểm khác nhau
- Mỗi phòng chiếu có thể chiếu nhiều phim khác nhau tại nhiều thời điểm khác nhau
- Mỗi một thời điểm nhất định, trong một phòng chiếu chỉ có duy nhất một phim được chiếu



Bài tập (2)

Thiết kế CSDL cho phần mềm quản lí bán vé cho một chuỗi rạp chiếu phim như trước, có bổ sung phần bán vé như sau:

- Mỗi vé in đầy đủ các thông tin: tên phim, phòng chiếu, giờ chiếu, số ghế, giá vé.
- Cùng một phim, chiếu tại cùng 1 phòng chiếu nhưng nếu ở các khung giờ và ngày khác nhau có thể có giá vé khác nhau.
- Mỗi khách hàng có thể mua nhiều vé của cùng suất chiếu hoặc khác suất chiếu và thanh toán 1 lần. Hóa đơn chứa đầy đủ thông tin số vé cho mỗi suất chiếu và tổng tiền thanh toán.
- Nhân viên chỉ bán vé cho khách hàng khi phòng chiếu tại giờ chiếu mà khách hàng yêu cầu vẫn còn đủ số lượng ghế trống cho khách hàng.
- Khách hàng có thể trả lại một số vé sau khi đã mua, và có thể phải chịu tiền phạt: trả trước 48h thì miễn phí, trả trước 24h thì mất phí 20%, trả trước 12h thì mất phí 40%, trả trước 6h thì mất phí 60%, trả sau 6h thì mất phí 100%, tính từ giờ khởi chiếu



Bài tập (3)

Thiết kế CSDL cho phần mềm quản lí bán vé cho một chuỗi rạp chiếu phim như trước, có bổ sung phần khách hàng thân thiết như sau:

- Mỗi khách hàng có một thẻ khách hàng thân thiết có thể tích điểm.
- Mỗi lần mua vé có xuất thẻ thì khách hàng được cộng điểm theo tỉ lệ: cứ 10000VND thì được cộng thêm 1 điểm. Ví dụ nếu hóa đơn thanh toán 116000VND thì được cộng 11 điểm.
- Khi số điểm đạt ngưỡng nào đấy thì có thể đổi vé xem phim miễn phí. Ví dụ, cứ 200 điểm được đổi 1 vé xem phim 2D, 400 điểm được đổi 1 vé xem phim 3D. Nếu khách hàng có 280 điểm và mua 2 vé xem phim với tổng hóa đơn là 200000VND, nếu khách hàng thanh toán hết thì sẽ được cộng 20 điểm vào thẻ. Nếu khách hàng muốn đổi vé miễn phí thì sẽ được đổi 1 vé, khách hàng chỉ phải trả 100000VND cho vé còn lại, trong thẻ còn 80 điểm, và sau giao dịch này khách hàng chỉ được cộng số điểm bằng tỉ lệ phần tiền thanh toán: 10 điểm ứng với 100000VND.



Dùng lệnh SQL



Insert

Thêm một khách sạn mới vào bảng tblHotel:

```
INSERT INTO `tblhotel`(`address`,`id`,`name`,`starLevel`)  
VALUES("Sai Gon",5,"Saigon Star",3);
```

id	name	address	starLevel	description
1	Daiwoo	Kim Mã, Hà Nội	5	NULL
2	Sofitel	Tây Hồ, Hà Nội	5	NULL
3	Metropole	Hoàn Kiếm, Hà Nội	4	NULL
4	Bảo Sơn	Cầu Giấy, Hà Nội	3	NULL
5	Saigon Star	Sai Gon	3	NULL
NULL	NULL	NULL	NULL	NULL



Update

Sửa một khách sạn đã có trong bảng tblHotel:

```
UPDATE `tblhotel`
```

```
SET `address` = "Quận 1, TP. Hồ Chí Minh"
```

```
WHERE id = 5;
```

id	name	address	starLevel	description
1	Daiwoo	Kim Mã, Hà Nội	5	NULL
2	Sofitel	Tây Hồ, Hà Nội	5	NULL
3	Metropole	Hoàn Kiếm, Hà Nội	4	NULL
4	Bảo Sơn	Cầu Giấy, Hà Nội	3	NULL
5	Saigon Star	Quận 1, TP. Hồ Chí Minh	3	NULL
NULL	NULL	NULL	NULL	NULL



Delete

Xóa một khách sạn trong bảng tblHotel:

```
DELETE FROM `tblhotel`
```

```
WHERE id = 5;
```

id	name	address	starLevel	description
1	Daiwoo	Kim Mã, Hà Nội	5	NULL
2	Sofitel	Tây Hồ, Hà Nội	5	NULL
3	Metropole	Hoàn Kiếm, Hà Nội	4	NULL
4	Bảo Sơn	Cầu Giấy, Hà Nội	3	NULL
NULL	NULL	NULL	NULL	NULL



Select (1)

Dữ liệu hiện tại của các bảng:

id	name	address	starLevel	description
1	Daiwoo	Kim Mã, Hà Nội	5	NULL
2	Sofitel	Tây Hồ, Hà Nội	5	NULL
3	Metropole	Hoàn Kiếm, Hà Nội	4	NULL
4	Bảo Sơn	Cầu Giấy, Hà Nội	3	NULL
5	Saigon Star	Quận 1, TP. Hồ Chí Minh	3	NULL

id	username	password	fullName	idCardNumber	idCardType	address	description
1	xuanhinh	abc123	Xuân Hinh	123456789	CMTND	Nam Định	NULL
2	minhvuong	abc123	Minh Vương	234567891	CMTND	Hà Nội	NULL
3	xuanbac	abc123	Xuân Bắc	345678912	CMTND	Hà Nội	NULL
4	tranthanh	abc123	Trần Thành	456789123	CMTND	Xì Gòn	NULL
5	david	abc123	David James	B0809076	Passport	Anh	NULL
6	tom	abc123	Tom Cruise	B123456	Passport	Úc	NULL
7	nhanvien	abc123	Nhân Viên	147258369	CMTND	KS	NULL

id	idHotel	name	type	displayPrice	description
1	4	102	single	700000	NULL
2	4	103	double	1000000	NULL
3	4	104	twink	1000000	NULL
4	4	202	single	650000	NULL
5	4	203	double	950000	NULL
6	4	204	twink	950000	NULL
7	4	302	single	900000	NULL
8	4	303	double	900000	NULL
9	4	304	twink	900000	NULL

id	idRoom	idUser	startDate	endDate	price
1	1	1	2013-08-25	2013-08-28	750000
2	1	2	2013-08-30	2013-09-02	800000
3	1	3	2013-09-07	2013-09-15	600000
4	4	4	2013-08-26	2013-08-30	750000
5	7	5	2013-08-28	2013-09-01	650000

id	idBooking	paymentDate	amount	paymentType	notes
1	2	2013-08-15	2400000	visa card	NULL
2	1	2013-08-28	2250000	cash	NULL
3	3	2013-09-07	2400000	cash	NULL
4	3	2013-09-15	2400000	master card	NULL



Select (2)

Tìm kiếm tất cả những người dùng có tên chứa chữ “a”:

```
SELECT * FROM tblUser
```

```
WHERE fullName LIKE "%a%";
```

id	username	password	fullName	idCardNumber	idCardType	address	description
1	xuanhinh	abc123	Xuân Hinh	123456789	CMTND	Nam Định	NULL
3	xuanbac	abc123	Xuân Bắc	345678912	CMTND	Hà Nội	NULL
4	tranthanh	abc123	Trần Thành	456789123	CMTND	Xi Gòn	NULL
5	david	abc123	David James	B0809076	Passport	Anh	NULL
7	nhanvien	abc123	Nhân Viên	147258369	CMTND	KS	NULL



Select (3)

Tìm kiếm tất cả những người dùng có đặt phòng từ 15/08/2013 đến 30/08/2013:

```
SELECT a.fullName, a.idCardNumber, a.idCardType, a.address,  
b.name AS `room`, b.type, c.startDate, c.endDate, c.price  
FROM tblUser a, tblRoom b, tblBooking c  
WHERE c.startDate BETWEEN "2013-08-15" AND "2013-08-30"  
AND c.endDate BETWEEN "2013-08-15" AND "2013-08-30"  
AND b.id = c.idRoom  
AND a.id = c.idUser;
```

fullName	idCardNumber	idCardType	address	room	type	startDate	endDate	price
Xuân Hình	123456789	CMTND	Nam Định	102	single	2013-08-25	2013-08-28	750000
Trần Thành	456789123	CMTND	Xi Gòn	202	single	2013-08-26	2013-08-30	750000



Select (4)

Tìm kiếm và sắp xếp những khách hàng đã trả tiền nhiều nhất đến ít nhất:

```
SELECT a.id, a.fullName, a.idCardNumber, a.idCardType, a.address,  
SUM(c.amount) AS `amount`
```

```
FROM tblUser a INNER JOIN tblBooking b ON b.idUser = a.id
```

```
INNER JOIN tblBill c ON c.idBooking = b.id
```

```
GROUP BY a.id, a.fullName, a.idCardNumber, a.idCardType, a.address  
ORDER BY `amount` DESC;
```

id	fullName	idCardNumber	idCardType	address	amount
3	Xuân Bắc	345678912	CMTND	Hà Nội	4800000
2	Minh Vương	234567891	CMTND	Hà Nội	2400000
1	Xuân Hình	123456789	CMTND	Nam Định	2250000



Bài tập (1)

1. Tìm kiếm và sắp xếp những khách hàng đã trả tiền nhiều nhất đến ít nhất trong khoảng thời gian từ ngày a đến ngày b?
2. Tìm kiếm và sắp xếp những phòng của một khách sạn X theo thứ tự tổng số tiền thu được (từ cao đến thấp) từ ngày a đến ngày b?
3. Tìm kiếm và sắp xếp những phòng của một khách sạn X theo thứ tự tổng số ngày có khách (từ cao đến thấp) tính từ ngày a đến ngày b?



Bài tập (2)

Với CSDL cho phần mềm quản lí bán vé cho một chuỗi rạp chiếu phim như trước, viết các lệnh SQL cho các chức năng sau:

- Thêm một phim mới vào CSDL
- Lên lịch chiếu phim trong một ngày cho một phòng chiếu
- Bán một số vé của một phim trong một phòng chiếu ở một khung giờ nào đó
- Tìm kiếm các phim + phòng chiếu + giờ chiếu trong một khoảng thời gian nào đó mà đang còn ghế trống
- Thống kê các phim có doanh thu cao nhất đến thấp nhất trong một khoảng thời gian nào đó
- Thống kê các khung giờ chiếu có đông khách hàng xem phim từ cao nhất đến thấp nhất
- Thống kê các khách hàng mua nhiều vé nhất đến ít vé nhất trong một khoảng thời gian nào đó



Kết nối DB bằng JDBC



Kết nối bằng JDBC (1)

```
public Connection getConnection(String dbClass, String dbUrl)
throws SQLException {
    Connection conn = null;

    try {
        Class.forName(dbClass);
        conn = DriverManager.getConnection (dbUrl);
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    } catch (SQLException e) {
        throws e;
    }
    return conn;
}
```

```
String dbClass = "com.mysql.jdbc.Driver";
String dbUrl =
    "jdbc:mysql://your.database.domain/yourDBname";
```



Kết nối bằng JDBC (2)

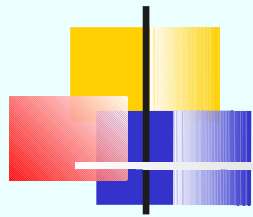
```
public Connection getConnection(String dbClass, String dbUrl, String
userName, String password) throws SQLException {
    Connection conn = null;

    try {
        Class.forName(dbClass);
        conn =
            DriverManager.getConnection (dbUrl, userName, password);
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    } catch (SQLException e) {
        throws e;
    }
    return conn;
}
```

```
String dbClass = "com.mysql.jdbc.Driver";
String dbUrl =
    "jdbc:mysql://your.database.domain/yourDBname";
```




Dùng lệnh SQL trong Java



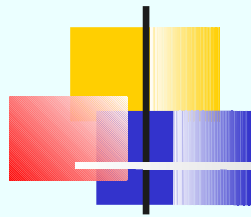
Dùng Statement (1)

```
String query = "Select * FROM users";
```

```
String query = "INSERT INTO users VALUES (« aaa », « bbb »)";
```

```
String query = "UPDATE users SET password = « xxx » WHERE id  
= 111";
```

```
String query = "DELETE FROM users WHERE id = 111";
```



Dùng Statement (2)

```
try {  
    Statement stmt = conn.createStatement();  
    ResultSet rs = stmt.executeQuery(query);  
  
} catch (ClassNotFoundException e) {  
    e.printStackTrace();  
} catch (SQLException e) {  
    e.printStackTrace();  
}
```



Dùng PreparedStatement

```
PreparedStatement updateSales = null;

String updateString = "update products " +
                      "set SALES = ? where ID = ?";

try {
    updateSales = conn.prepareStatement(updateString);

    updateSales.setInt(1, value);
    updateSales.setInt(2, productId);
    updateSales.executeUpdate();
} catch (SQLException e) {
    throw e;
}
```



Dùng StoreProcedure (1)

```
String createProcedure =  
    "create procedure GET_MAX_OF_SALE(IN productId int,  
        OUT value int) " +  
    "begin " +  
        "select MAX(value) into productValue " +  
        "from products " +  
        "where ID = productId; " +  
        "select productValue; " +  
    "end";  
  
try {  
    Statement stmt = conn.createStatement();  
    stmt.executeUpdate(createProcedure);  
} catch (SQLException e) {  
    throw e;  
}
```



Dùng StoreProcedure (2)

```
try {  
    CallableStatement cs =  
        conn.prepareCall("{call GET_MAX_OF_SALE(?,?) }");  
  
    cs.setInt(1, productId);  
    cs.registerOutParameter(2, Types.INT);  
    cs.executeQuery();  
  
    int maxValue = cs.getInt(2);  
  
} catch (SQLException e) {  
    throw e;  
}
```



Lấy dữ liệu ra



Dữ liệu từ ResultSet (1)

```
String query = "Select * FROM users";

try {
    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery(query);

    while (rs.next()) {
        System.out.println(rs.getString(1));
    }
} catch (SQLException e) {
    e.printStackTrace();
}
```



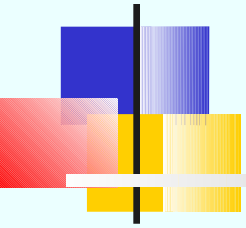

Dữ liệu từ ResultSet (2)

```
String query = "Select * FROM users";

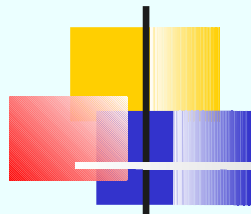
try {
    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery(query);

    // get number of row in resultSet
    int rowcount = 0;
    if (rs.last()) {
        rowcount = rs.getRow();
        rs.beforeFirst(); // not rs.first()
    }

    while (rs.next()) {
        // do something with data...
    }
} catch (SQLException e) {
    e.printStackTrace();
}
```



Làm việc với Transaction



Điều khiển chế độ commit (1)

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

    ....

    conn.commit();
} catch (SQLException e) {
    if (conn != null) {
        try {
            conn.rollback();
        } catch (SQLException excep) {
            throw excep;
        }
    }
    throw e;
} finally {
    conn.setAutoCommit(true);
}
}
```



Điều khiển chế độ commit (2)

```
public void updateSales(int productId, int value) throws  
SQLException {
```

```
    PreparedStatement updateSales = null;  
    PreparedStatement updateTotal = null;
```

```
    String updateString = "update products " +  
        "set SALES = ? where ID = ?";
```

```
    String updateStatement = "update totalSale " +  
        "set TOTAL = TOTAL + ? where productId = ?";
```

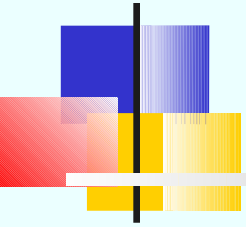
```
    try {  
        conn.setAutoCommit(false);  
        updateSales = conn.prepareStatement(updateString);  
        updateTotal = conn.prepareStatement(updateStatement);  
  
        updateSales.setInt(1, value);  
        updateSales.setInt(2, transactionId);  
        updateSales.executeUpdate();
```



Điều khiển chế độ commit (3)

```
        updateTotal.setInt(1, value);
        updateTotal.setInt(2, productId);
        updateTotal.executeUpdate();
        conn.commit();
    } catch (SQLException e) {
        if (conn != null) {
            try {
                conn.rollback();
            } catch (SQLException excep) {
                throw excep;
            }
        }
        throw e;
    } finally {
        if (updateSales != null) { updateSales.close(); }
        if (updateTotal != null) { updateTotal.close(); }
        conn.setAutoCommit(true);
    }
}
```

Ví dụ: Bài toán quản lí đặt phòng khách sạn





Bài toán

- Một khách sạn (id, tên, địa chỉ, số sao, mô tả) có nhiều phòng (id, hạng phòng, giá niêm yết, mô tả)
- Mỗi phòng có thể được đặt bởi nhiều khách hàng (id, tên, số id, kiểu thẻ id, địa chỉ, mô tả) tại nhiều thời điểm khác nhau
- Mỗi khách hàng có thể đặt nhiều phòng tại nhiều thời điểm khác nhau nhưng chỉ ở 1 phòng tại 1 thời điểm nhất định, xác định 1 giá xác định
- Khách hàng chỉ có thể đặt phòng nếu phòng còn trống trong suốt thời gian khách hàng muốn đặt
- Khi trả phòng, nhân viên in phiếu thanh toán bao gồm tên khách sạn, tên khách hàng, số phòng, hạng phòng, ngày đến, ngày đi và tổng số tiền thanh toán
- Khách hàng có thể thanh toán nhiều lần cho đến trước ngày trả phòng
- Nếu hủy đặt phòng sẽ bị phạt, tùy vào thời hạn hủy sớm hay muộn



Yêu cầu

- Thiết kế các bảng CSDL cho bài toán
- Định nghĩa các lớp thực thể cho bài toán
- Định nghĩa phương thức thêm một phòng khách sạn vào CSDL
- Định nghĩa phương thức tìm kiếm danh sách phòng trống trong một khoảng thời gian xác định
- Định nghĩa phương thức cho phép 1 khách hàng đặt một phòng trong một thời gian xác định
- Định nghĩa phương thức tính doanh thu của khách sạn trong một thời gian xác định
- Định nghĩa phương thức liệt kê danh sách các phòng có tỉ lệ khách hàng đặt cao nhất trong năm



Thiết kế CSDL (1)

- Trước hết cần có 3 bảng:

tblHotel: id, name, starLevel, address, description

tblRoom: id, name, type, displayPrice, description

tblUser: id, fullName, username, password, idCardNumber, idCardType, address, description

Nhận xét:

- 1 khách sạn có nhiều phòng, 1 phòng chỉ ở trong 1 khách sạn → quan hệ giữa bảng tblHotel và tblRoom là 1-n
 - 1 khách hàng đặt nhiều phòng, 1 phòng cũng có nhiều người đặt → quan hệ giữa tblRoom và tblUser là n-n, → phải chuẩn hóa
- Tạo thêm một bảng đặt chỗ chi tiết **tblBooking**: idRoom, idUser, startDate, endDate, price, description
- quan hệ giữa tblRoom và tblBooking là 1-n (một phòng có nhiều lần đặt), và giữa tblUser và tblBooking cũng là 1-n (một người có nhiều lần đặt)



Thiết kế CSDL (2)

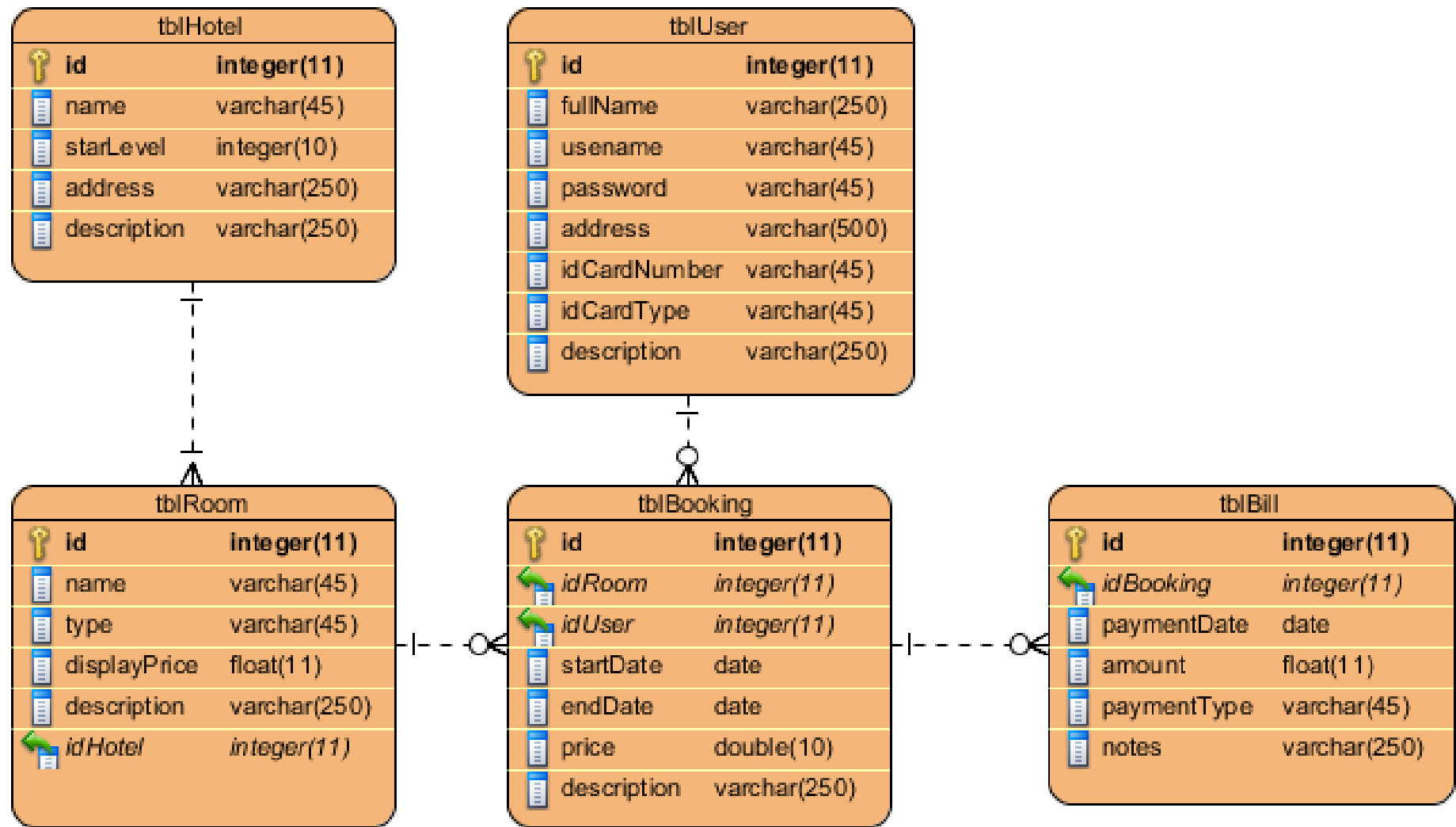
- Thêm 1 bảng:

tblBill: id, idBooking, paymentDate, amount, paymentType, notes

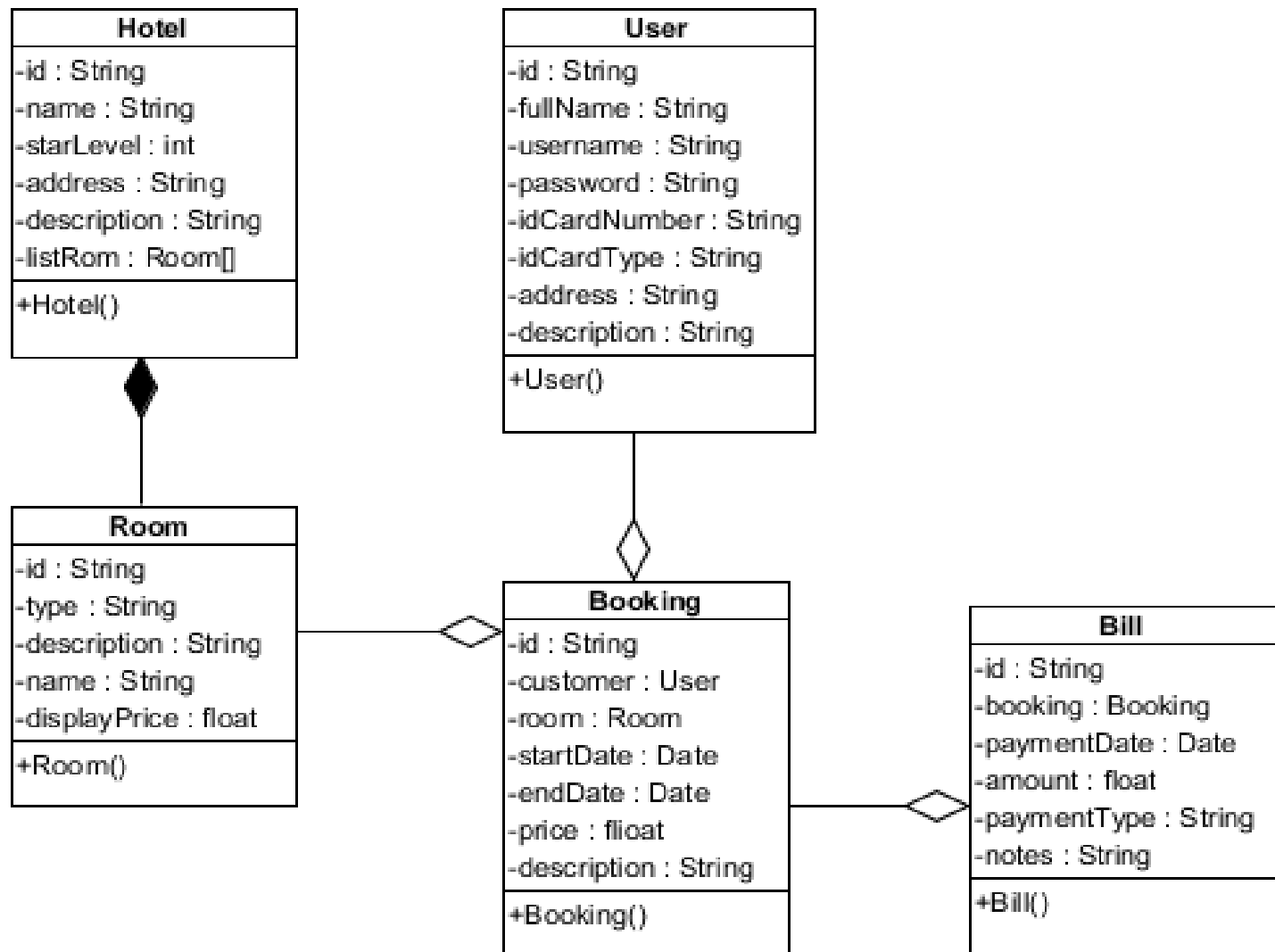
Nhận xét:

- 1 đặt phòng có thể trả tiền nhiều lần (đặt cọc, trả trước, thanh toán khi ở xong) → quan hệ giữa bảng tblBooking và tblBill là 1-n

Thiết kế CSDL (3)



Các lớp thực thể





Thêm phòng

```
public void addRoom(int hotelID, Room r) throws SQLException{
    String sql = "INSERT INTO tblRoom(name, idHotel, type,
displayPrice, description) VALUES(?,?,?,?)" ;
    PreparedStatement prstm = null;

    try {
        prstm = conn.prepareStatement(sql);

        prstm.setString(1, r.getName());
        prstm.setInt(2, hotelID);
        prstm.setString(3, r.getType());
        prstm.setFloat(4, r.getDisplayPrice());
        prstm.setString(5, r.getDescription());
        prstm.executeUpdate();
    } catch (SQLException e ) {
        throw e;
    }
}
```



Tìm phòng trống (1)

```
String sql = "SELECT * FROM tblRoom WHERE id NOT IN (SELECT RoomId  
from tblBooking WHERE ((StartDate BETWEEN ? AND ?) OR EndDate BETWEEN  
? AND ?) OR (? BETWEEN StartDate AND EndDate) OR (? BETWEEN StartDate  
AND EndDate));
```



Tìm phòng trống (2)

```
public Room[] searchFreeRoom(Date sd, Date ed) throws SQLException{
    Room[] result = null;
    String sql = <slide trước>
    try {
        PreparedStatement prstm = conn.prepareStatement(sql);
        prstm.setDate(1, sd); prstm.setDate(3, sd); prstm.setDate(5, sd);
        prstm.setDate(2, ed); prstm.setDate(4, ed); prstm.setDate(6, ed);

        ResultSet rs = prstm.executeQuery();
        if (rs.last()) {
            result = new Room[rs.getRow()];
            rs.beforeFirst();
        }
        int count = 0;
        while (rs.next()) {
            result[count] = new Room(rs.getInt(1), rs.getString(2),
                                     rs.getString(3), rs.getFloat(4), rs.getString(5));
            count++;
        }
    } catch (SQLException e) { throw e;}
    return result;
}
```



Đặt phòng

```
public void bookRoom(Booking b) throws SQLException{
    String sql = INSERT INTO tblBooking(idRoom, idUser, startDate,
endDate, price, description) VALUES(?, ?, ?, ?, ?, ?)";
    try {
        PreparedStatement prstm = conn.prepareStatement(sql);

        prstm.setInt(1, b.getRoom().getId());
        prstm.setInt(2, b.getUser().getId());
        prstm.setDate(3, b.getStartDate());
        prstm.setDate(4, b.getEndDate());
        prstm.setFloat(5, b.getPrice());
        prstm.setString(6, b.getDescription());
        prstm.executeUpdate();

    } catch (SQLException e) {
        throw e;
    }
}
```




Tính doanh thu

```
public double totalIncomeByPeriod(Date sd, Date ed) throws
SQLException{
    double result = 0;
    String sql = SELECT SUM(b.amount)FROM tblBooking a INNER JOIN
tblBill b ON b.idBooking = a.id WHERE ((a.startDate BETWEEN ? AND
?) AND (a.endDate BETWEEN ? AND ?))" ;
    try {
        PreparedStatement prstm = conn.prepareStatement(sql);

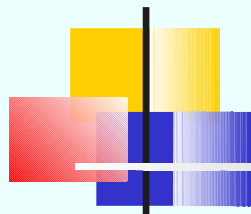
        prstm.setDate(1, sd); prstm.setDate(3, sd);
        prstm.setDate(2, ed); prstm.setDate(4, ed);
        ResultSet rs = prstm.executeQuery();

        if(rs.next()){
            result = rs.getDouble(1);
        }
    } catch (SQLException e ) {
        throw e;
    }
    return result;
}
```



Tìm phòng đặt nhiều

```
public Room[] searchHotRoom() throws SQLException{
    Room[] result = null;
    String sql = "SELECT a.* FROM tblRoom a ORDER BY (SELECT
SUM(DATEDIFF(DAY, b.startDate, b.endDate)) FROM tblBooking b WHERE
b.roomId = a.id) DESC";
    try {
        PreparedStatement prstm = conn.prepareStatement(sql);
        ResultSet rs = prstm.executeQuery();
        if (rs.last()) {
            result = new Room[rs.getRow()];
            rs.beforeFirst();
        }
        int count = 0;
        while (rs.next()) {
            result[count] = new Room(rs.getInt(1), rs.getString(2),
rs.getString(3), rs.getFloat(4), rs.getString(5))
            count++;
        }
    } catch (SQLException e) {
        throw e;
    }
    return result;
}
```



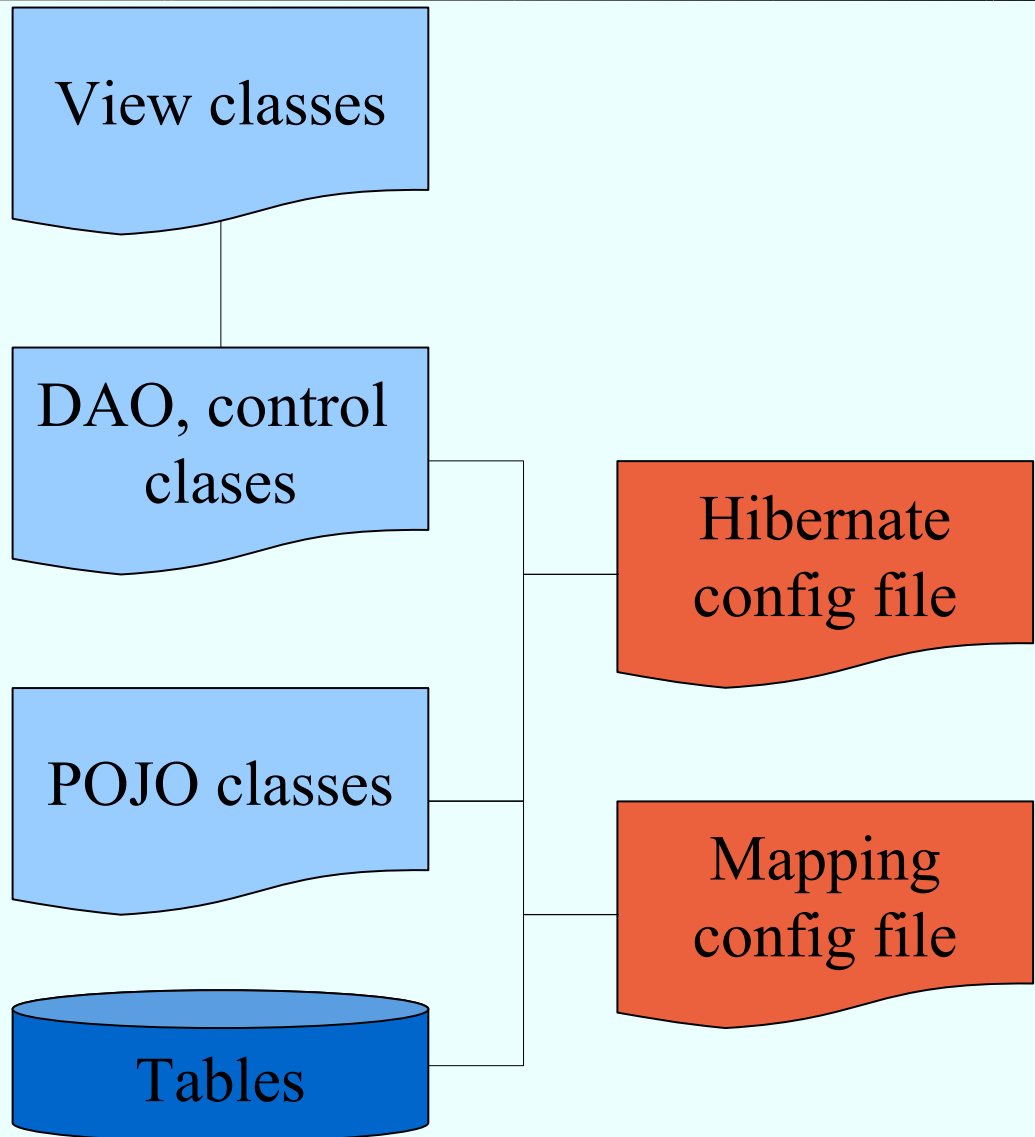
Bài tập

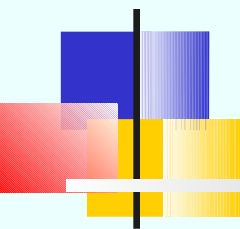
- Cài đặt tầng giao diện cho bài toán
- Tích hợp các phương thức đã định nghĩa vào để được ứng dụng hoàn chỉnh



Dùng Hibernate

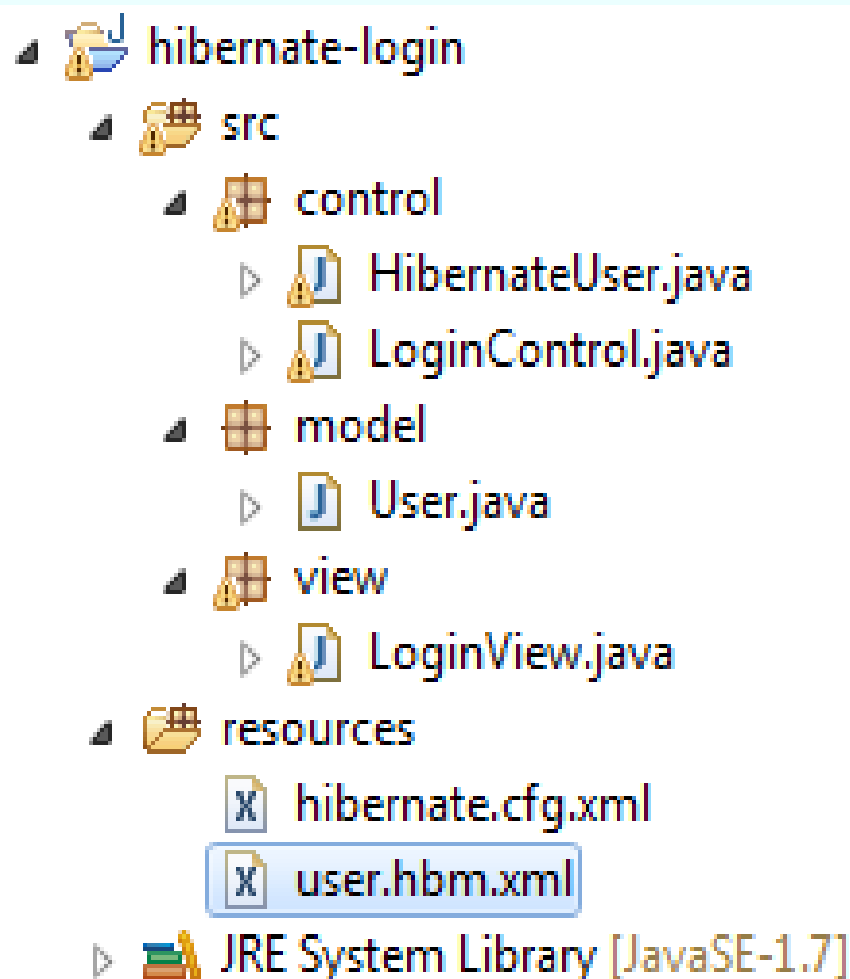
Mô hình sử dụng Hibernate














Ví dụ: Login dùng Hibernate

Cấu trúc file của project



Bảng User

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
 id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
 username	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 password	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 fullName	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 idCardNumber	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 idCardType	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 address	VARCHAR(250)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 description	VARCHAR(500)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 address	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL



File Hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
  <session-factory>
    <property name="connection.driver_class"> com.mysql.jdbc.Driver </property>
    <property name="connection.url">jdbc:mysql://localhost:3306/hotelmanagement
      </property>
    <property name="connection.username">root</property>
    <property name="connection.password">12345678</property>
    <property name="connection.pool_size">1</property>
    <property name="dialect">org.hibernate.dialect.MySQLDialect </property>
    <property name="current_session_context_class">thread</property>
    <property name="cache.provider_class"> org.hibernate.cache.NoCacheProvider
      </property>
    <property name="show_sql">true</property>
    <property name="hbm2ddl.auto">update</property>
    <mapping resource="user.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```



Lớp POJO User

```
public class User implements Serializable{
    private static final long serialVersionUID = 1L;
    private Integer id;
    private String username;
    private String password;
    private String fullName;
    private String idCardNumber;
    private String idCardType;
    private String address;
    private String description;

    public User(){

    }

    public get/set() { // cho tất cả các thuộc tính
    }
}
```



File User.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
  <class name="model.User" table="tbluser" >
    <id name="id" type="int" column="id" >
      <generator class="native"/>
    </id>
    <property name="username"> <column name="username" /> </property>
    <property name="password"> <column name="password"/> </property>
    <property name="fullName"> <column name="fullName" /> </property>
    <property name="idCardNumber"> <column name="idCardNumber"/>
    </property>
    <property name="idCardType"> <column name="idCardType" />
    </property>
    <property name="address"> <column name="address"/> </property>
    <property name="description"> <column name="description" />
    </property>
  </class>
</hibernate-mapping>
```



Lớp HibernateUser(1)

```
public class HibernateUser {  
    private static final SessionFactory sessionFactory =  
        buildSessionFactory();  
  
    private static SessionFactory buildSessionFactory() {  
        try {  
            // Create the SessionFactory from hibernate.cfg.xml  
            return new AnnotationConfiguration().configure()  
                .buildSessionFactory();  
        } catch (Throwable ex) {  
            System.err.println("Initial SessionFactory creation failed."  
                + ex);  
            throw new ExceptionInInitializerError(ex);  
        }  
    }  
  
    public static SessionFactory getSessionFactory() {  
        return sessionFactory;  
    }  
}
```



Lớp HibernateUser(2)

```
public boolean checkLogin(User user){
    boolean result = false;
    Session session = this.getSessionFactory().getCurrentSession();
    session.beginTransaction();
    List<User> users = null;
    String sql = "FROM User U WHERE U.username = :username
                  AND U.password = :password";
    try {
        Query q = session.createQuery(sql);
        q.setParameter("username", user.getUsername());
        q.setParameter("password", user.getPassword());
        users = (List<User>)q.list();
        if (users.size() > 0) result = true;
    } catch (HibernateException e) {
        e.printStackTrace();
        session.getTransaction().rollback();
    }
    session.getTransaction().commit();
    return result;
}
```



Lớp LoginView(1)

```
public class LoginView extends JFrame implements ActionListener{
    private JTextField txtUsername;
    private JPasswordField txtPassword;
    private JButton btnLogin;
    private User model;

    public LoginView(){
        super("Login MVC using Hibernate");
        txtUsername = new JTextField(15);
        txtPassword = new JPasswordField(15);
        txtPassword.setEchoChar('*');
        btnLogin = new JButton("Login");
        JPanel content = new JPanel();
        content.setLayout(new FlowLayout());
        content.add(new JLabel("Username:"));
        content.add(txtUsername);
        content.add(new JLabel("Password:"));
        content.add(txtPassword);
        content.add(btnLogin);
        btnLogin.addActionListener(this);
        this.setContentPane(content);
        this.pack();
        this.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e){
                System.exit(0);
            }
        });
    }
}
```



Lớp LoginView(2)

```
public void actionPerformed(ActionEvent e) {  
  
    public User getUser(){  
        model = new User();  
        model.setUsername(txtUsername.getText());  
        model.setPassword(txtPassword.getText());  
        return model;  
    }  
  
    public void showMessage(String msg){  
        JOptionPane.showMessageDialog(this, msg);  
    }  
  
    public void addLoginListener(ActionListener log) {  
        btnLogin.addActionListener(log);  
    }  
}
```



Lớp LoginControl(1)

```
public class LoginControl {
    private User model;
    private LoginView view;

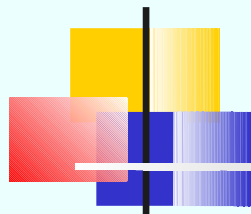
    public LoginControl(LoginView view){
        this.view = view;
        view.addLoginListener(new LoginListener());
    }

    class LoginListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            try {
                model = view.getUser();
                HibernateUser hu = new HibernateUser();
                if(hu.checkLogin(model)){
                    view.showMessage("Login succesfully!");
                }else{
                    view.showMessage("Invalid username and/or
password!");
                }
            } catch (Exception ex) {
                view.showMessage(ex.getStackTrace().toString());
            }
        }
    }
}
```




Lớp LoginControl(2)

```
public static void main(String[] args) {  
    LoginView view          = new LoginView();  
    LoginControl controller = new LoginControl(view);  
    view.setVisible(true);  
}  
}
```



Bài tập

- Cài đặt bài toán quản lí khách sạn bằng hibernate



Questions?
