实验 02：操作关系型数据库

一、实验目的

1、了解完整的关系型数据库应用系统的基本构成。

2、了解基本的软件分层架构思想：通过 Model（模型）、DAO（数据访问对象）、Service（服务）、Main（主程序）的包结构。

3、掌握 MySQL 数据库的基本操作（增删改查）

4、掌握使用 Java（JDBC）操作数据库的标准流程

5、理解关系型数据库的设计范式与实体关系

二、实验学时

2 学时

三、实验类型

综合性

四、实验需求

1、硬件

每人配备计算机 1 台，建议优先使用个人计算机开展实验。

实验基于信息技术学院教学容器化云计算平台开展。

2、软件

安装 IntelliJ IDEA Community。

MySQL 8.0。

项目管理使用 GitLab。

3、网络

本地主机能够访问互联网和实验中心网络。

4、工具

无

五、实验任务

1、使用 Java 实现的医院信息管理系统，包含了对 MySQL 数据库的增删改查和多表关联查询操作。

六、实验内容及步骤

1、表结构设计

(1) **创建科室表**

(2) **创建医生表**

(3) **创建患者表**

(4) **创建病房表**

(5) **创建预约表**

(6) **创建住院记录表**

SQL

```sql
-- 创建数据库
CREATE DATABASE IF NOT EXISTS hospital_db;
USE hospital_db;

-- 科室表
CREATE TABLE departments (
    dept_id INT PRIMARY KEY AUTO_INCREMENT,
    dept_name VARCHAR(50) NOT NULL,
    location VARCHAR(100),
    head_doctor_id INT
);

-- 医生表
CREATE TABLE doctors (
    doctor_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    specialization VARCHAR(100),
    dept_id INT,
    phone_number VARCHAR(20),
    email VARCHAR(100),
    hire_date DATE,
    FOREIGN KEY (dept_id) REFERENCES departments(dept_id)
);

-- 患者表
CREATE TABLE patients (
    patient_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    date_of_birth DATE,
    gender ENUM('男', '女', '其他'),
    phone_number VARCHAR(20),
    address TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- 病房表
CREATE TABLE wards (
    ward_id INT PRIMARY KEY AUTO_INCREMENT,
    ward_name VARCHAR(50) NOT NULL,
    dept_id INT,
```

```
    capacity INT,
    current_occupancy INT DEFAULT 0,
    FOREIGN KEY (dept_id) REFERENCES departments(dept_id)
);

-- 预约表
CREATE TABLE appointments (
    appointment_id INT PRIMARY KEY AUTO_INCREMENT,
    patient_id INT,
    doctor_id INT,
    appointment_date DATETIME,
    status ENUM('预约中', '已完成', '已取消'),
    notes TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (patient_id) REFERENCES patients(patient_id),
    FOREIGN KEY (doctor_id) REFERENCES doctors(doctor_id)
);

-- 住院记录表
CREATE TABLE hospitalizations (
    hospitalization_id INT PRIMARY KEY AUTO_INCREMENT,
    patient_id INT,
    ward_id INT,
    doctor_id INT,
    admission_date DATE,
    discharge_date DATE NULL,
    diagnosis TEXT,
    treatment_plan TEXT,
    FOREIGN KEY (patient_id) REFERENCES patients(patient_id),
    FOREIGN KEY (ward_id) REFERENCES wards(ward_id),
    FOREIGN KEY (doctor_id) REFERENCES doctors(doctor_id)
);

-- 添加外键约束到科室表
ALTER TABLE departments ADD FOREIGN KEY (head_doctor_id) REFERENCES doc
tors(doctor_id);
```

2、初始化数据

SQL

```sql
-- 插入科室数据
INSERT INTO departments (dept_name, location) VALUES
('内科', '一楼东侧'),
('外科', '二楼西侧'),
('儿科', '三楼南侧'),
('妇产科','二楼东侧'),
('骨科', '一楼西侧');

-- 插入医生数据
INSERT INTO doctors (first_name, last_name, specialization, dept_id, phone_number, email, hire_date) VALUES
('张', '小明', '心脏病学', 1, '13800138001', 'zhangxiaoming@qq.com', '2015-03-10'),
('李', '中华', '神经内科', 1, '13800138002', 'lizhonghua@qq.com', '2017-06-15'),
('王', '成刚', '普通外科', 2, '13800138003', 'wangchenggang@qq.com', '2010-08-20'),
('赵', '敏敏', '儿科医学', 3, '13800138004', 'zhaominmin@qq.com', '2018-02-05'),
('刘', '艳芳', '妇产科学', 4, '13800138005', 'liuyanfang@qq.com', '2012-11-30'),
('陈', '国强', '骨科医学', 5, '13800138006', 'chenguoqiang@qq.com', '2016-07-22');

-- 更新科室表设置主任医生
UPDATE departments SET head_doctor_id = 1 WHERE dept_id = 1;
UPDATE departments SET head_doctor_id = 3 WHERE dept_id = 2;
UPDATE departments SET head_doctor_id = 4 WHERE dept_id = 3;
UPDATE departments SET head_doctor_id = 5 WHERE dept_id = 4;
UPDATE departments SET head_doctor_id = 6 WHERE dept_id = 5;

-- 插入患者数据
INSERT INTO patients (first_name, last_name, date_of_birth, gender, phone_number, address) VALUES
('陈', '小明', '1985-07-12', '男', '13900139001', '河南省郑州市金水区100号'),
('杨', '国华', '1978-11-23', '女', '13900139002', '河南省郑州市金水区200号'),
('黄', '晓明', '2015-03-15', '女', '13900139003', '河南省郑州市金水区300号'),
('吴', '敏', '1992-09-05', '男', '13900139004', '河南省郑州市金水区400号'),
('郑', '海波', '1980-12-30', '女', '13900139005', '河南省郑州市金水区500
```

```
号');

-- 插入病房数据
INSERT INTO wards (ward_name, dept_id, capacity) VALUES
('内科病房1', 1, 4),
('内科病房2', 1, 6),
('外科病房1', 2, 5),
('儿科病房1', 3, 8),
('妇产科病房1', 4, 6),
('骨科病房1', 5, 4);

-- 插入预约数据
INSERT INTO appointments (patient_id, doctor_id, appointment_date, status) VALUES
(1, 1, '2023-10-15 09:00:00', '已完成'),
(2, 3, '2023-10-16 10:30:00', '预约中'),
(3, 4, '2023-10-17 14:00:00', '预约中'),
(4, 6, '2023-10-18 11:15:00', '已取消'),
(5, 5, '2023-10-19 15:45:00', '预约中');

-- 插入住院记录数据
INSERT INTO hospitalizations (patient_id, ward_id, doctor_id, admission_date, diagnosis, treatment_plan) VALUES
(1, 1, 1, '2023-10-10', '高血压', '药物治疗和定期监测'),
(3, 4, 4, '2023-10-12', '肺炎', '抗生素治疗和休息'),
(5, 5, 5, '2023-10-14', '孕期检查', '定期产检和营养指导');
```

3、构建 Java 项目

(1) 数据查询测试用例

SQL

```sql
-- 查询所有医生信息
SELECT * FROM doctors;

-- 查询特定科室的医生
SELECT d.doctor_id, d.first_name, d.last_name, d.specialization, dept.dept_name
FROM doctors d
JOIN departments dept ON d.dept_id = dept.dept_id
WHERE dept.dept_name = '内科';

-- 查询患者及其预约信息
SELECT p.first_name, p.last_name, d.first_name AS doctor_first_name,
       d.last_name AS doctor_last_name, a.appointment_date, a.status
FROM appointments a
JOIN patients p ON a.patient_id = p.patient_id
JOIN doctors d ON a.doctor_id = d.doctor_id;

-- 查询当前住院患者信息
SELECT p.first_name, p.last_name, w.ward_name, d.first_name AS doctor_first_name,
       d.last_name AS doctor_last_name, h.admission_date, h.diagnosis
FROM hospitalizations h
JOIN patients p ON h.patient_id = p.patient_id
JOIN wards w ON h.ward_id = w.ward_id
JOIN doctors d ON h.doctor_id = d.doctor_id
WHERE h.discharge_date IS NULL;

-- 查询各科室病房使用情况
SELECT dept.dept_name, w.ward_name, w.capacity, w.current_occupancy,
       (w.capacity - w.current_occupancy) AS available_beds
FROM wards w
JOIN departments dept ON w.dept_id = dept.dept_id;
```

(2) 数据修改测试用例

SQL

```sql
-- 更新医生电话号码
UPDATE doctors SET phone_number = '13800138000' WHERE doctor_id = 1;

-- 更新患者地址
UPDATE patients SET address = '河南省郑州市金水区建国路200号' WHERE patient_id = 1;

-- 完成预约
UPDATE appointments SET status = '已完成'
WHERE appointment_id = 2;

-- 患者出院
UPDATE hospitalizations SET discharge_date = CURDATE()
WHERE hospitalization_id = 1;

-- 更新病房占用情况
UPDATE wards SET current_occupancy = current_occupancy - 1
WHERE ward_id = (SELECT ward_id FROM hospitalizations WHERE hospitalization_id = 1);
```

(3) 数据删除测试用例

SQL

```sql
-- 取消预约
DELETE FROM appointments WHERE appointment_id = 4;

-- 删除患者记录(需要先删除相关记录)
-- 首先删除相关预约记录
DELETE FROM appointments WHERE patient_id = 4;
-- 然后删除患者
DELETE FROM patients WHERE patient_id = 4;
```

(4) 复杂查询测试用例

SQL

```sql
-- 查询各科室医生数量
SELECT d.dept_name, COUNT(doc.doctor_id) AS doctor_count
FROM departments d
LEFT JOIN doctors doc ON d.dept_id = doc.dept_id
GROUP BY d.dept_name
ORDER BY doctor_count DESC;

-- 查询每位医生的患者数量
SELECT doc.doctor_id, doc.first_name, doc.last_name,
       COUNT(DISTINCT a.patient_id) AS patient_count
FROM doctors doc
LEFT JOIN appointments a ON doc.doctor_id = a.doctor_id
GROUP BY doc.doctor_id, doc.first_name, doc.last_name
ORDER BY patient_count DESC;

-- 查询今天之后的预约
SELECT p.first_name AS patient_first_name, p.last_name AS patient_last_name,
       doc.first_name AS doctor_first_name, doc.last_name AS doctor_last_name,
       a.appointment_date, a.status
FROM appointments a
JOIN patients p ON a.patient_id = p.patient_id
JOIN doctors doc ON a.doctor_id = doc.doctor_id
WHERE a.appointment_date > NOW()
ORDER BY a.appointment_date;

-- 查询各科室病房使用率
SELECT dept.dept_name, w.ward_name, w.capacity, w.current_occupancy,
       ROUND((w.current_occupancy * 100.0 / w.capacity), 2) AS usage_rate
FROM wards w
JOIN departments dept ON w.dept_id = dept.dept_id
ORDER BY usage_rate DESC;

-- 查询患者的完整住院历史
SELECT p.first_name, p.last_name,
       h.admission_date, h.discharge_date,
       w.ward_name,
       doc.first_name AS doctor_first_name, doc.last_name AS doctor_last_name,
       h.diagnosis
```

```
39 FROM hospitalizations h
40 JOIN patients p ON h.patient_id = p.patient_id
41 JOIN wards w ON h.ward_id = w.ward_id
42 JOIN doctors doc ON h.doctor_id = doc.doctor_id
43 ORDER BY h.admission_date DESC;
```

(5) 创建 Java 项目

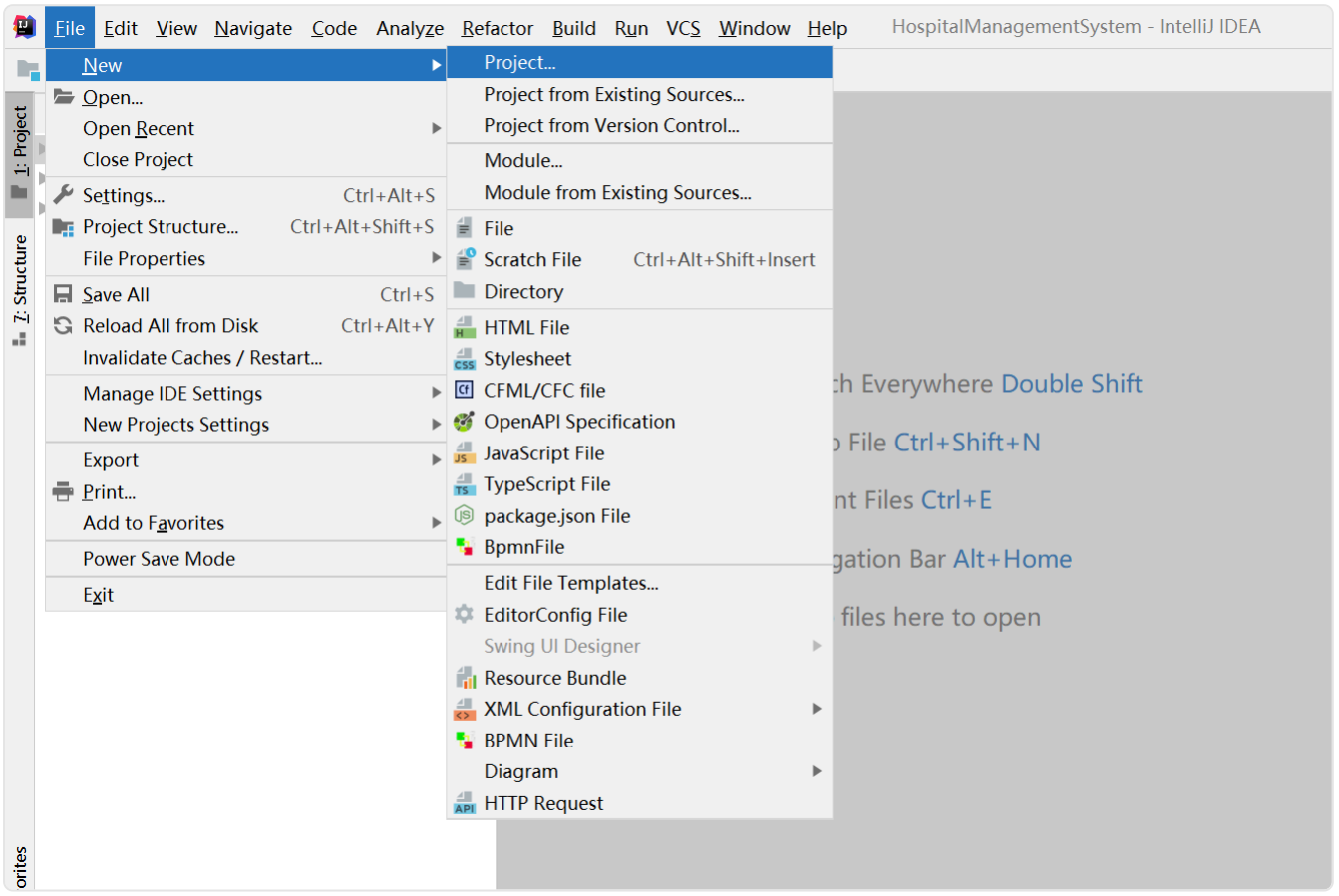

图 6-3-5-1 创建 java 项目

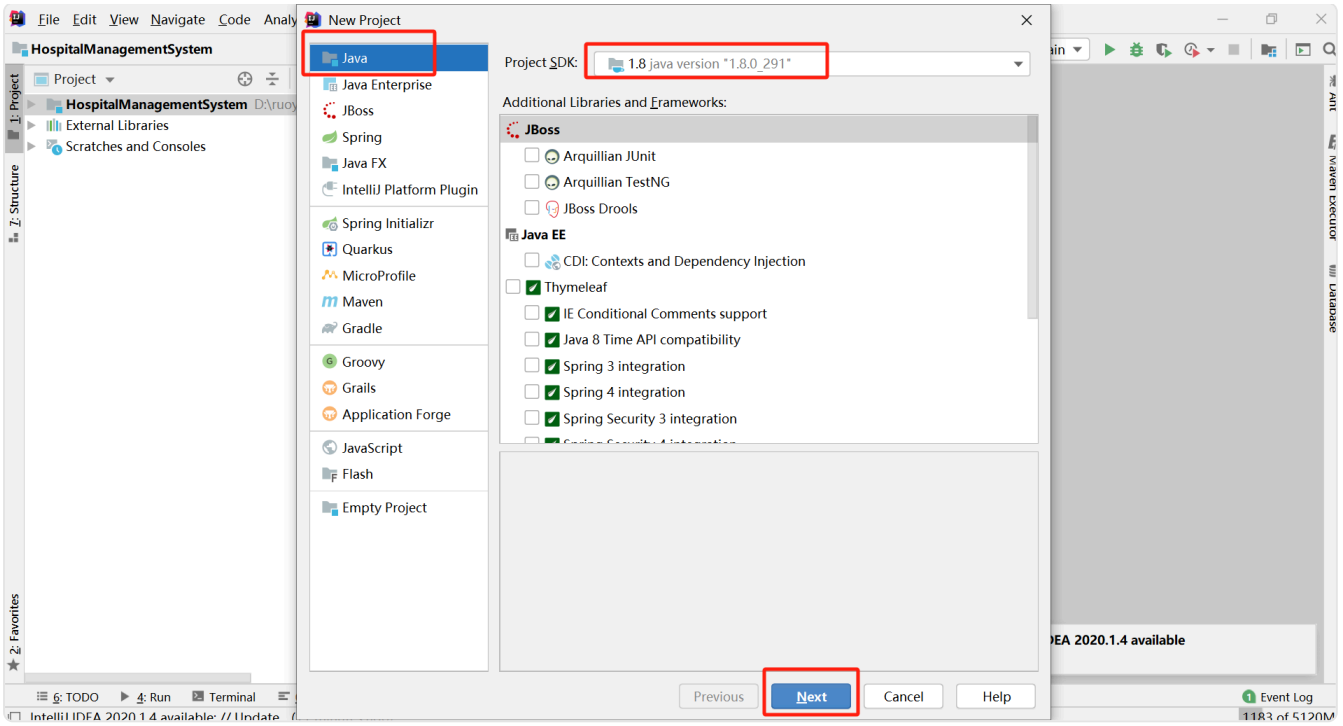

图 6-3-5-2 选择 java 项目并指定 JDK

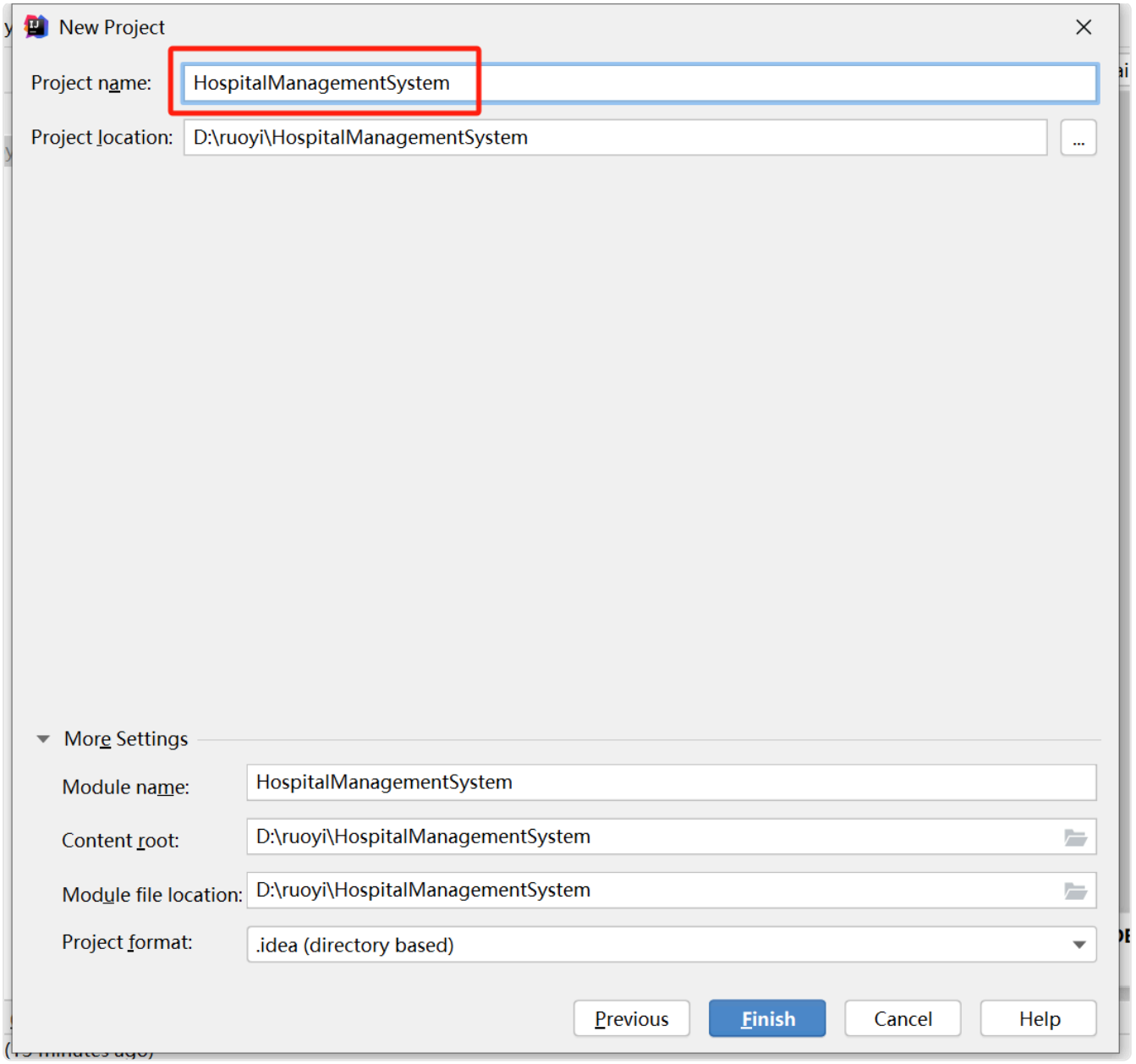

图 6-3-5-3 指定项目名称

(6) 导入 mysql 数据库驱动

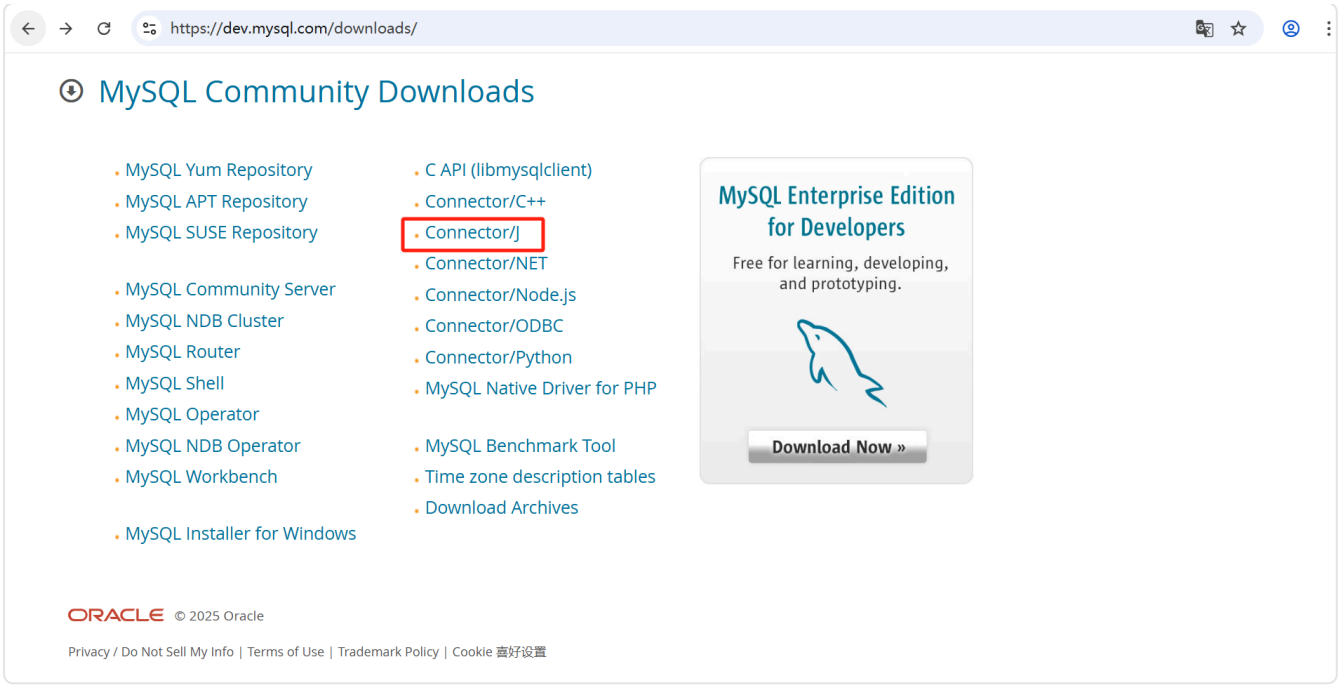

图 6-3-6-1 下载 mysql 数据库官方驱动

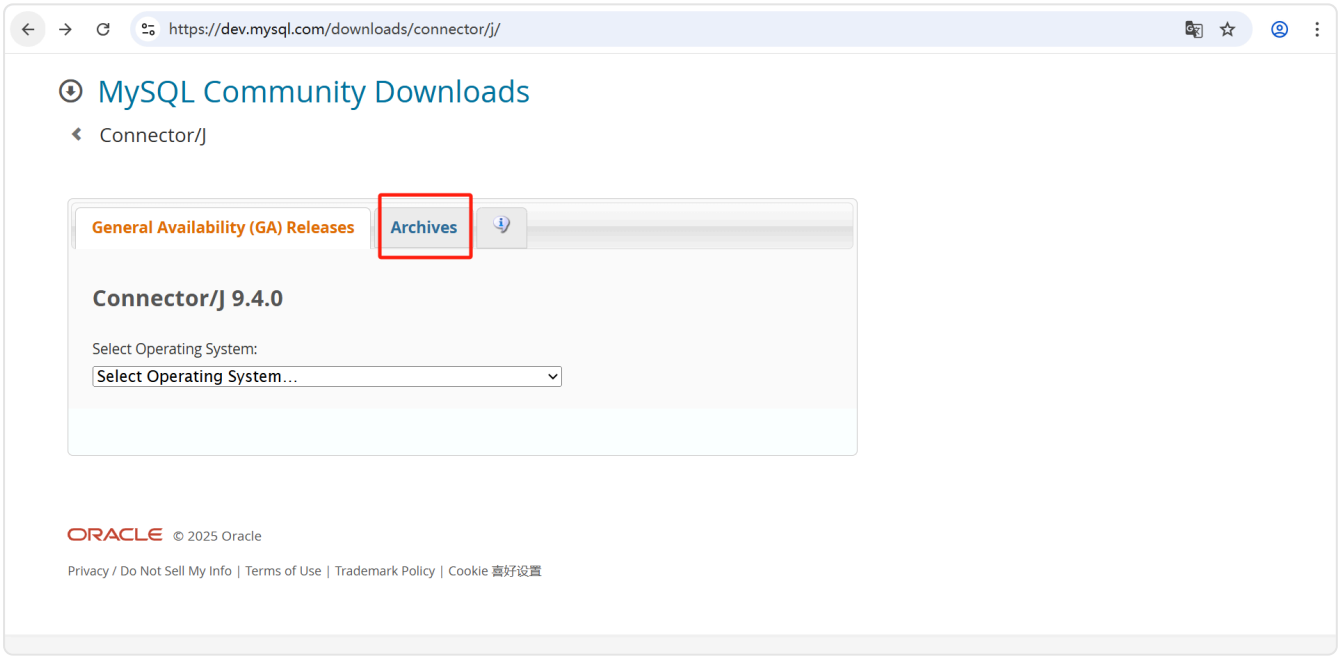

图 6-3-6-2 选择 archives

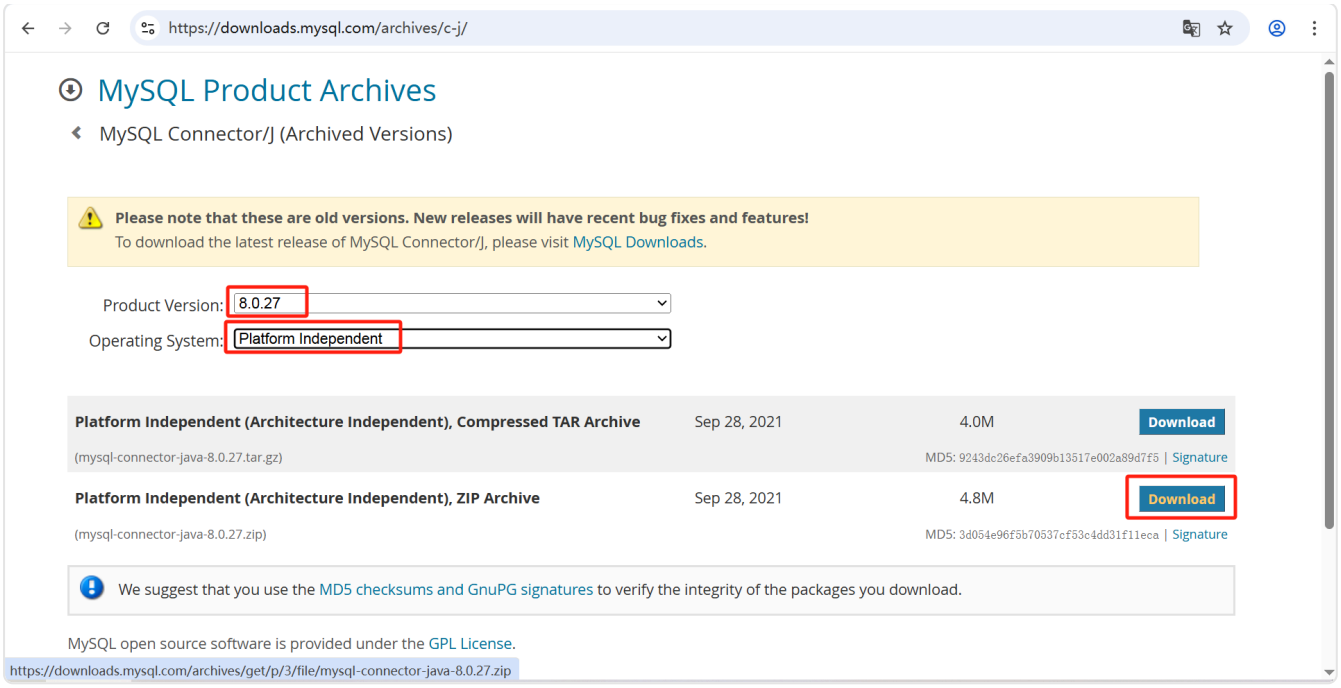

图 6-3-6-3 指定驱动版本和操作系统类型

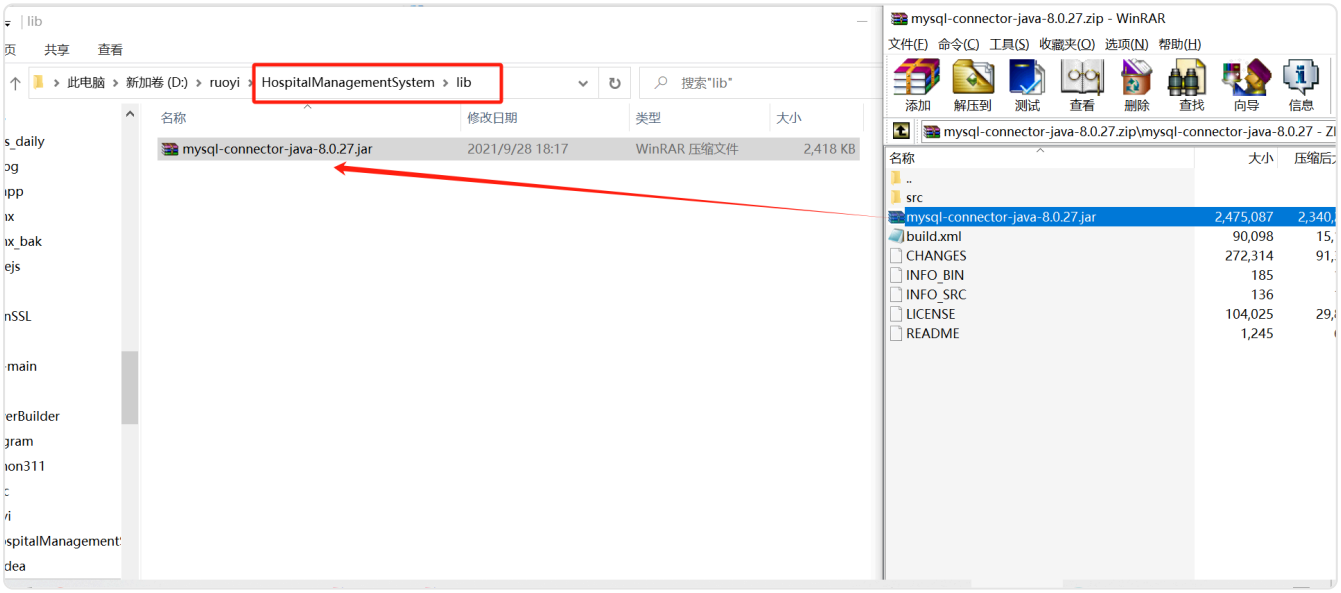

图 6-3-6-4 项目 src 同级路径下创建 lib 目录并将驱动文件拖动至该目录

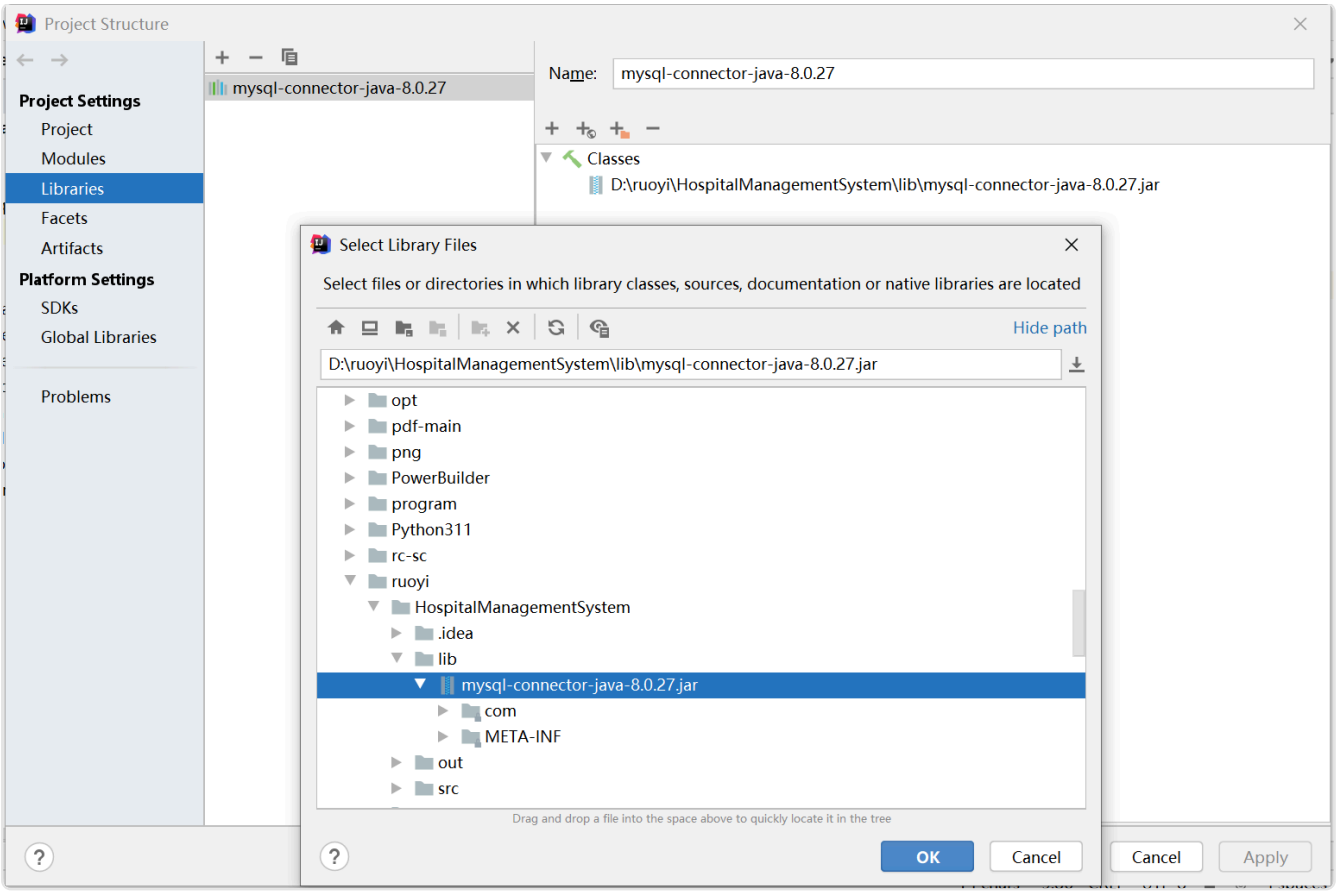

图 6-3-6-5 在项目中引入 jar 包

(7) 创建 Java 类

Java

```
HospitalManagementSystem/
├── src/
│   ├── model/
│   │   ├── Department.java
│   │   ├── Doctor.java
│   │   ├── Patient.java
│   │   ├── Ward.java
│   │   ├── Appointment.java
│   │   └── Hospitalization.java
│   ├── dao/
│   │   ├── DatabaseConnection.java
│   │   ├── DepartmentDAO.java
│   │   ├── DoctorDAO.java
│   │   ├── PatientDAO.java
│   │   ├── WardDAO.java
│   │   ├── AppointmentDAO.java
│   │   └── HospitalizationDAO.java
│   ├── service/
│   │   └── HospitalService.java
│   └── Main.java
└── lib/
    └── mysql-connector-java-8.0.27.jar
```

项目结构

Java

```java
// DatabaseConnection.java
package dao;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/hospital_db";
    private static final String USER = "username";
    private static final String PASSWORD = "password";

    static {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

数据库连接工具类

Java

```java
public class Doctor {
    private int doctorId;
    private String firstName;
    private String lastName;
    private String specialization;
    private int deptId;
    private String phoneNumber;
    private String email;
    private String hireDate;

    // 构造方法、getter和setter省略，但实际需要实现
}
```

Doctor 类（实体类）

Java

```java
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class DoctorDAO {

    // 插入医生记录
    public void addDoctor(Doctor doctor) throws SQLException {
        String sql = "INSERT INTO doctors (first_name, last_name, specialization, dept_id, phone_number, email, hire_date) VALUES (?, ?, ?, ?, ?, ?, ?)";
        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, doctor.getFirstName());
            pstmt.setString(2, doctor.getLastName());
            pstmt.setString(3, doctor.getSpecialization());
            pstmt.setInt(4, doctor.getDeptId());
            pstmt.setString(5, doctor.getPhoneNumber());
            pstmt.setString(6, doctor.getEmail());
            pstmt.setString(7, doctor.getHireDate());
            pstmt.executeUpdate();
        }
    }

    // 根据ID查询医生
    public Doctor getDoctorById(int doctorId) throws SQLException {
        String sql = "SELECT * FROM doctors WHERE doctor_id = ?";
        Doctor doctor = null;
        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, doctorId);
            ResultSet rs = pstmt.executeQuery();
            if (rs.next()) {
                doctor = new Doctor();
                doctor.setDoctorId(rs.getInt("doctor_id"));
                doctor.setFirstName(rs.getString("first_name"));
                doctor.setLastName(rs.getString("last_name"));
                doctor.setSpecialization(rs.getString("specialization"));
                doctor.setDeptId(rs.getInt("dept_id"));
                doctor.setPhoneNumber(rs.getString("phone_number"));
                doctor.setEmail(rs.getString("email"));
```

```
40                 doctor.setHireDate(rs.getString("hire_date"));
41             }
42         }
43         return doctor;
44     }
45 
46     // 查询所有医生
47     public List<Doctor> getAllDoctors() throws SQLException {
48         List<Doctor> doctors = new ArrayList<>();
49         String sql = "SELECT * FROM doctors";
50         try (Connection conn = DatabaseConnection.getConnection();
51              Statement stmt = conn.createStatement();
52              ResultSet rs = stmt.executeQuery(sql)) {
53             while (rs.next()) {
54                 Doctor doctor = new Doctor();
55                 doctor.setDoctorId(rs.getInt("doctor_id"));
56                 doctor.setFirstName(rs.getString("first_name"));
57                 doctor.setLastName(rs.getString("last_name"));
58                 doctor.setSpecialization(rs.getString("specializatio
   n"));
59                 doctor.setDeptId(rs.getInt("dept_id"));
60                 doctor.setPhoneNumber(rs.getString("phone_number"));
61                 doctor.setEmail(rs.getString("email"));
62                 doctor.setHireDate(rs.getString("hire_date"));
63                 doctors.add(doctor);
64             }
65         }
66         return doctors;
67     }
68 
69     // 更新医生信息
70     public void updateDoctor(Doctor doctor) throws SQLException {
71         String sql = "UPDATE doctors SET first_name = ?, last_name =
   ?, specialization = ?, dept_id = ?, phone_number = ?, email = ?, hire_d
   ate = ? WHERE doctor_id = ?";
72         try (Connection conn = DatabaseConnection.getConnection();
73              PreparedStatement pstmt = conn.prepareStatement(sql)) {
74             pstmt.setString(1, doctor.getFirstName());
75             pstmt.setString(2, doctor.getLastName());
76             pstmt.setString(3, doctor.getSpecialization());
77             pstmt.setInt(4, doctor.getDeptId());
78             pstmt.setString(5, doctor.getPhoneNumber());
79             pstmt.setString(6, doctor.getEmail());
80             pstmt.setString(7, doctor.getHireDate());
```

```
            pstmt.setInt(8, doctor.getDoctorId());
            pstmt.executeUpdate();
        }
    }

    // 删除医生
    public void deleteDoctor(int doctorId) throws SQLException {
        String sql = "DELETE FROM doctors WHERE doctor_id = ?";
        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, doctorId);
            pstmt.executeUpdate();
        }
    }
}
```

DoctorDAO 类（数据访问对象）

类似地，需要为其他表（患者、科室、预约等）创建 DAO 类，示例中省略。

Java

```java
public List<Doctor> getDoctorsByDepartment(String deptName) throws SQLException {
    List<Doctor> doctors = new ArrayList<>();
    String sql = "SELECT d.* FROM doctors d JOIN departments dept ON d.dept_id = dept.dept_id WHERE dept.dept_name = ?";
    try (Connection conn = DatabaseConnection.getConnection();
         PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setString(1, deptName);
        ResultSet rs = pstmt.executeQuery();
        while (rs.next()) {
            Doctor doctor = new Doctor();
            doctor.setDoctorId(rs.getInt("doctor_id"));
            doctor.setFirstName(rs.getString("first_name"));
            doctor.setLastName(rs.getString("last_name"));
            doctor.setSpecialization(rs.getString("specialization"));
            doctor.setDeptId(rs.getInt("dept_id"));
            doctor.setPhoneNumber(rs.getString("phone_number"));
            doctor.setEmail(rs.getString("email"));
            doctor.setHireDate(rs.getString("hire_date"));
            doctors.add(doctor);
        }
    }
    return doctors;
}
```

多表关联查询核心示例

Java

```java
// Department.java
package model;

public class Department {
    private int deptId;
    private String deptName;
    private String location;
    private int headDoctorId;

    // 构造方法
    public Department() {}

    public Department(int deptId, String deptName, String location, int headDoctorId) {
        this.deptId = deptId;
        this.deptName = deptName;
        this.location = location;
        this.headDoctorId = headDoctorId;
    }

    // Getter和Setter方法
    public int getDeptId() { return deptId; }
    public void setDeptId(int deptId) { this.deptId = deptId; }

    public String getDeptName() { return deptName; }
    public void setDeptName(String deptName) { this.deptName = deptName; }

    public String getLocation() { return location; }
    public void setLocation(String location) { this.location = location; }

    public int getHeadDoctorId() { return headDoctorId; }
    public void setHeadDoctorId(int headDoctorId) { this.headDoctorId = headDoctorId; }

    @Override
    public String toString() {
        return "Department [deptId=" + deptId + ", deptName=" + deptName +
                ", location=" + location + ", headDoctorId=" + headDoctorId + "]";
```

```
37     }
38 }
```

Department.java

Java

```java
// Doctor.java
package model;

import java.util.Date;

public class Doctor {
    private int doctorId;
    private String firstName;
    private String lastName;
    private String specialization;
    private int deptId;
    private String phoneNumber;
    private String email;
    private Date hireDate;

    // 构造方法、Getter和Setter方法
    public Doctor() {}

    public Doctor(int doctorId, String firstName, String lastName, String specialization,
                  int deptId, String phoneNumber, String email, Date hireDate) {
        this.doctorId = doctorId;
        this.firstName = firstName;
        this.lastName = lastName;
        this.specialization = specialization;
        this.deptId = deptId;
        this.phoneNumber = phoneNumber;
        this.email = email;
        this.hireDate = hireDate;
    }

    // Getter和Setter方法
    public int getDoctorId() { return doctorId; }
    public void setDoctorId(int doctorId) { this.doctorId = doctorId; }

    public String getFirstName() { return firstName; }
    public void setFirstName(String firstName) { this.firstName = firstName; }

    public String getLastName() { return lastName; }
    public void setLastName(String lastName) { this.lastName = lastNam
```

```
    e; }
40
41      public String getSpecialization() { return specialization; }
42      public void setSpecialization(String specialization) { this.special
    ization = specialization; }
43
44      public int getDeptId() { return deptId; }
45      public void setDeptId(int deptId) { this.deptId = deptId; }
46
47      public String getPhoneNumber() { return phoneNumber; }
48      public void setPhoneNumber(String phoneNumber) { this.phoneNumber
    = phoneNumber; }
49
50      public String getEmail() { return email; }
51      public void setEmail(String email) { this.email = email; }
52
53      public Date getHireDate() { return hireDate; }
54      public void setHireDate(Date hireDate) { this.hireDate = hireDate;
    }
55
56      @Override
57      public String toString() {
58          return "Doctor [doctorId=" + doctorId + ", firstName=" + firstN
    ame +
59                  ", lastName=" + lastName + ", specialization=" + special
    ization +
60                  ", deptId=" + deptId + ", phoneNumber=" + phoneNumber +
61                  ", email=" + email + ", hireDate=" + hireDate + "]";
62      }
63  }
```

Doctor.java

其他实体类（Patient, Ward, Appointment,Hospitalization 等）实现类似，这里省略。

Java

```java
// DepartmentDAO.java
package dao;

import model.Department;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class DepartmentDAO {

    // 添加科室
    public boolean addDepartment(Department department) {
        String sql = "INSERT INTO departments (dept_name, location, hea
d_doctor_id) VALUES (?, ?, ?)";

        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {

            pstmt.setString(1, department.getDeptName());
            pstmt.setString(2, department.getLocation());
            pstmt.setInt(3, department.getHeadDoctorId());

            int affectedRows = pstmt.executeUpdate();
            return affectedRows > 0;

        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    // 获取所有科室
    public List<Department> getAllDepartments() {
        List<Department> departments = new ArrayList<>();
        String sql = "SELECT * FROM departments";

        try (Connection conn = DatabaseConnection.getConnection();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {

            while (rs.next()) {
                Department department = new Department();
```

```
                department.setDeptId(rs.getInt("dept_id"));
                department.setDeptName(rs.getString("dept_name"));
                department.setLocation(rs.getString("location"));
                department.setHeadDoctorId(rs.getInt("head_doctor_i
d"));

                departments.add(department);
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }

        return departments;
    }

    // 根据ID获取科室
    public Department getDepartmentById(int deptId) {
        String sql = "SELECT * FROM departments WHERE dept_id = ?";
        Department department = null;

        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {

            pstmt.setInt(1, deptId);
            ResultSet rs = pstmt.executeQuery();

            if (rs.next()) {
                department = new Department();
                department.setDeptId(rs.getInt("dept_id"));
                department.setDeptName(rs.getString("dept_name"));
                department.setLocation(rs.getString("location"));
                department.setHeadDoctorId(rs.getInt("head_doctor_i
d"));
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }

        return department;
    }

    // 更新科室信息
```

```java
    public boolean updateDepartment(Department department) {
        String sql = "UPDATE departments SET dept_name = ?, location = ?, head_doctor_id = ? WHERE dept_id = ?";

        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {

            pstmt.setString(1, department.getDeptName());
            pstmt.setString(2, department.getLocation());
            pstmt.setInt(3, department.getHeadDoctorId());
            pstmt.setInt(4, department.getDeptId());

            int affectedRows = pstmt.executeUpdate();
            return affectedRows > 0;

        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    // 删除科室
    public boolean deleteDepartment(int deptId) {
        String sql = "DELETE FROM departments WHERE dept_id = ?";

        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {

            pstmt.setInt(1, deptId);

            int affectedRows = pstmt.executeUpdate();
            return affectedRows > 0;

        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }
}
```

DepartmentDAO.java

其他 DAO 类（DoctorDAO, PatientDAO, WardDAO, AppointmentDAO, HospitalizationDAO 等）实现类似，这里省略。

Java

```java
// HospitalService.java
package service;

import dao.*;
import model.*;
import java.util.List;

public class HospitalService {
    private DepartmentDAO departmentDAO = new DepartmentDAO();
    private DoctorDAO doctorDAO = new DoctorDAO();
    private PatientDAO patientDAO = new PatientDAO();
    private WardDAO wardDAO = new WardDAO();
    private AppointmentDAO appointmentDAO = new AppointmentDAO();
    private HospitalizationDAO hospitalizationDAO = new HospitalizationDAO();

    // 科室相关操作
    public boolean addDepartment(Department department) {
        return departmentDAO.addDepartment(department);
    }

    public List<Department> getAllDepartments() {
        return departmentDAO.getAllDepartments();
    }

    public Department getDepartmentById(int deptId) {
        return departmentDAO.getDepartmentById(deptId);
    }

    public boolean updateDepartment(Department department) {
        return departmentDAO.updateDepartment(department);
    }

    public boolean deleteDepartment(int deptId) {
        return departmentDAO.deleteDepartment(deptId);
    }

    // 医生相关操作
    public boolean addDoctor(Doctor doctor) {
        return doctorDAO.addDoctor(doctor);
    }

```

```java
    public List<Doctor> getAllDoctors() {
        return doctorDAO.getAllDoctors();
    }

    public Doctor getDoctorById(int doctorId) {
        return doctorDAO.getDoctorById(doctorId);
    }

    public List<Doctor> getDoctorsByDepartment(int deptId) {
        return doctorDAO.getDoctorsByDepartment(deptId);
    }

    public boolean updateDoctor(Doctor doctor) {
        return doctorDAO.updateDoctor(doctor);
    }

    public boolean deleteDoctor(int doctorId) {
        return doctorDAO.deleteDoctor(doctorId);
    }

    // 多表关联查询示例
    public List<Doctor> getDoctorsWithDepartmentInfo() {
        return doctorDAO.getDoctorsWithDepartmentInfo();
    }

    public List<Appointment> getAppointmentsWithPatientAndDoctorInfo()
{
        return appointmentDAO.getAppointmentsWithPatientAndDoctorInfo
();
    }

    public List<Hospitalization> getCurrentHospitalizations() {
        return hospitalizationDAO.getCurrentHospitalizations();
    }

    // 其他方法...
}
```

HospitalService.java

Java

```java
// Main.java
import service.HospitalService;
import model.*;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;

public class Main {
    private static HospitalService hospitalService = new HospitalService();
    private static SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");

    public static void main(String[] args) {
        try {
            // 1. 添加科室
            Department dept1 = new Department(0, "心脏内科", "一楼东侧", 0);
            hospitalService.addDepartment(dept1);

            // 2. 添加医生
            Doctor doctor1 = new Doctor(0, "张", "小明", "心脏病学", 1,
                                        "13800138001", "zhangming@hospital.com",
                                        dateFormat.parse("2015-03-10"));
            hospitalService.addDoctor(doctor1);

            // 3. 添加患者
            Patient patient1 = new Patient(0, "陈", "国华", dateFormat.parse("1985-07-12"),
                                           "男", "13900139001", "郑州市人民路100号", new Date());
            hospitalService.addPatient(patient1);

            // 4. 查询所有科室
            System.out.println("所有科室:");
            List<Department> departments = hospitalService.getAllDepartments();
            for (Department dept : departments) {
                System.out.println(dept);
            }

```

```
36              // 5. 查询所有医生
37              System.out.println("\n所有医生:");
38              List<Doctor> doctors = hospitalService.getAllDoctors();
39              for (Doctor doctor : doctors) {
40                  System.out.println(doctor);
41              }
42
43              // 6. 多表关联查询：获取医生及其科室信息
44              System.out.println("\n医生及其科室信息:");
45              List<Doctor> doctorsWithDept = hospitalService.getDoctorsWi
    thDepartmentInfo();
46              for (Doctor doctor : doctorsWithDept) {
47                  System.out.println(doctor.getFirstName() + " " + docto
    r.getLastName() +
48                                     " - " + doctor.getSpecialization() +
49                                     " - 科室: " + doctor.getDeptId()); //
    这里可以扩展为显示科室名称
50              }
51
52              // 7. 多表关联查询：获取当前住院患者信息
53              System.out.println("\n当前住院患者:");
54              List<Hospitalization> hospitalizations = hospitalService.ge
    tCurrentHospitalizations();
55              for (Hospitalization hospitalization : hospitalizations) {
56                  System.out.println("患者ID: " + hospitalization.getPatie
    ntId() +
57                                     ", 病房ID: " + hospitalization.getWardI
    d() +
58                                     ", 入院日期: " + hospitalization.getAdmi
    ssionDate());
59              }
60
61          } catch (Exception e) {
62              e.printStackTrace();
63          }
64      }
65  }
```

主程序 Main.java 示例

Java

```java
// DoctorDAO.java 中的多表查询方法
public List<Doctor> getDoctorsWithDepartmentInfo() {
    List<Doctor> doctors = new ArrayList<>();
    String sql = "SELECT d.*, dept.dept_name " +
                 "FROM doctors d " +
                 "JOIN departments dept ON d.dept_id = dept.dept_id";

    try (Connection conn = DatabaseConnection.getConnection();
         Statement stmt = conn.createStatement();
         ResultSet rs = stmt.executeQuery(sql)) {

        while (rs.next()) {
            Doctor doctor = new Doctor();
            doctor.setDoctorId(rs.getInt("doctor_id"));
            doctor.setFirstName(rs.getString("first_name"));
            doctor.setLastName(rs.getString("last_name"));
            doctor.setSpecialization(rs.getString("specialization"));
            doctor.setDeptId(rs.getInt("dept_id"));
            doctor.setPhoneNumber(rs.getString("phone_number"));
            doctor.setEmail(rs.getString("email"));
            doctor.setHireDate(rs.getDate("hire_date"));

            // 可以扩展Doctor类添加departmentName字段
            doctors.add(doctor);
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }

    return doctors;
}
```

多表关联查询主要方法实现示例

Java

```java
// HospitalizationDAO.java 中的多表查询方法
public List<Hospitalization> getCurrentHospitalizations() {
    List<Hospitalization> hospitalizations = new ArrayList<>();
    String sql = "SELECT h.*, p.first_name, p.last_name, w.ward_name, d.first_name AS doc_first_name, d.last_name AS doc_last_name " +
                 "FROM hospitalizations h " +
                 "JOIN patients p ON h.patient_id = p.patient_id " +
                 "JOIN wards w ON h.ward_id = w.ward_id " +
                 "JOIN doctors d ON h.doctor_id = d.doctor_id " +
                 "WHERE h.discharge_date IS NULL";

    try (Connection conn = DatabaseConnection.getConnection();
         Statement stmt = conn.createStatement();
         ResultSet rs = stmt.executeQuery(sql)) {

        while (rs.next()) {
            Hospitalization hospitalization = new Hospitalization();
            hospitalization.setHospitalizationId(rs.getInt("hospitalization_id"));
            hospitalization.setPatientId(rs.getInt("patient_id"));
            hospitalization.setWardId(rs.getInt("ward_id"));
            hospitalization.setDoctorId(rs.getInt("doctor_id"));
            hospitalization.setAdmissionDate(rs.getDate("admission_date"));
            hospitalization.setDischargeDate(rs.getDate("discharge_date"));
            hospitalization.setDiagnosis(rs.getString("diagnosis"));
            hospitalization.setTreatmentPlan(rs.getString("treatment_plan"));

            // 可以扩展Hospitalization类添加这些字段
            hospitalizations.add(hospitalization);
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }

    return hospitalizations;
}
```

HospitalizationDAO.java 中的多表查询主要方法实现示例

七、实验考核

1、本课程实验考核方案

本课程实验考核采用【实验智能评】【实验随堂查】方式开展，根据不同的实验内容选择不同的考核方式。

【实验智能评】：实验完成后提交 GitLab，通过自动化代码评审工具进行评分。

【实验随堂查】：在实验课上通过现场演示的方式向实验指导教师进行汇报，并完成现场问答交流。

2、本实验考核要求

本实验考核方式：实验智能评

实验 1-3 作为本课程第 1 次实验考核。

考核要求：

（1）学生通过 GitLab 提交实验成果：{此部分说明需要提交的内容}。

（2）由 GitLab 根据成果和交流情况综合评分。