1、综合案例：数据表与简单 Java 类（多对多）

2、具体内容

定义一个学生选课的操作表：三张数据表：

- 学生表：学生编号、姓名、年龄；
- 课程表：课程编号、课程名称、学分；
- 学生-课程关系表：学生编号、课程编号、成绩。

要求，可以实现如下的信息输出：

- 可以找到一门课程，以及参加此课程的所有的学生信息，和他的成绩；
- 可以根据一个学生，找到所参加的所有课程和每门课程的成绩。

1、 定义出基本类，暂时不考虑所有的关系。

```
class Student {
    private int stuid ;
    private String name ;
    private int age ;
    public Student() {}
    public Student(int stuid,String name,int age) {
        this.stuid = stuid ;
        this.name = name ;
```

```
        this.age = age ;
    }
    public String getInfo() {
        return "【STUDENT】学生编号：" + this.stuid + "，姓名：" + this.name + "，年龄：" + this.age ;
    }
}
class Course {
    private int cid ;
    private String name ;
    private int credit ;
    public Course() {}
    public Course(int cid,String name,int credit) {
        this.cid = cid ;
        this.name = name ;
        this.credit = credit ;
    }
    public String getInfo() {
        return "【COURSE】课程编号：" + this.cid + "，名称：" + this.name + "，学分：" + this.credit ;
    }
}
```

2、 一个学生有多门课，一门课有多个学生，所应该互相保存有各自的对象数组。

```
class Student {
    private int stuid ;
    private String name ;
    private int age ;
    private Course courses [] ;
    public Student() {}
    public Student(int stuid,String name,int age) {
        this.stuid = stuid ;
        this.name = name ;
        this.age = age ;
    }
    public void setCourses(Course [] courses) {
        this.courses = courses ;
    }
    public Course [] getCourses() {
        return this.courses ;
    }
    public String getInfo() {
        return "【STUDENT】学生编号：" + this.stuid + "，姓名：" + this.name + "，年龄：" + this.age ;
    }
}
class Course {
    private int cid ;
```

```
    private String name ;
    private int credit ;
    private Student [] students ;
    public Course() {}
    public Course(int cid,String name,int credit) {
        this.cid = cid ;
        this.name = name ;
        this.credit = credit ;
    }
    public void setStudents(Student [] students) {
        this.students = students ;
    }
    public Student[] getStudents() {
        return this.students ;
    }
    public String getInfo() {
        return "【COURSE】课程编号：" + this.cid + "，名称：" + this.name + "，学分：" + this.credit ;
    }
}
```

3、 学生和每门课程之间都会有一个成绩。现在发现关系表里面不光有关系字段还有一个普通字段，那么应该再建立一个类。

```
class StudentCourse {  // 学生选课
    private Student student ;
    private Course course ;
    private double score ;
    public StudentCourse() {}
    public StudentCourse(Student student,Course course,double score) {
        this.student = student ;
        this.course = course ;
        this.score = score ;
    }
    public Student getStudent() {
        return this.student ;
    }
    public Course getCourse() {
        return this.course ;
    }
    public double getScore() {
        return this.score ;
    }
}
```

```
class Student {
    private int stuid ;
    private String name ;
    private int age ;
```

```
    private StudentCourse studentCourses [] ;
    public Student() {}
    public Student(int stuid,String name,int age) {
        this.stuid = stuid ;
        this.name = name ;
        this.age = age ;
    }
    public void setStudentCourses(StudentCourse studentCourses []) {
        this.studentCourses = studentCourses ;
    }
    public StudentCourse [] getStudentCourses() {
        return this.studentCourses ;
    }
    public String getInfo() {
        return "【STUDENT】学生编号：" + this.stuid + "，姓名：" + this.name + "，年龄：" + this.age ;
    }
}
class Course {
    private int cid ;
    private String name ;
    private int credit ;
    private StudentCourse studentCourses [] ;
    public Course() {}
    public Course(int cid,String name,int credit) {
        this.cid = cid ;
        this.name = name ;
        this.credit = credit ;
    }
    public void setStudentCourses(StudentCourse studentCourses []) {
        this.studentCourses = studentCourses ;
    }
    public StudentCourse [] getStudentCourses() {
        return this.studentCourses ;
    }
    public String getInfo() {
        return "【COURSE】课程编号：" + this.cid + "，名称：" + this.name + "，学分：" + this.credit ;
    }
}
```

4、 要进行操作的实现

```
public class TestDemo {
    public static void main(String args[]) {
        // 第一步：根据结构进行关系的设置
        // 1、创建各自的独立对象
        Student stu1 = new Student(1,"张三",18) ;
```

```
        Student stu2 = new Student(2,"李四",19) ;
        Student stu3 = new Student(3,"王五",18) ;
        Course ca = new Course(1001,"马克思主义哲学",3) ;
        Course cb = new Course(1002,"操作系统",2) ;
        // 2、需要设置学生和课程的关系，这里面需要准备出成绩
        stu1.setStudentCourses(new StudentCourse[] {
                new StudentCourse(stu1,ca,99.9),
                new StudentCourse(stu1,cb,87.0)
        } ) ;
        stu2.setStudentCourses(new StudentCourse[] {
                new StudentCourse(stu2,ca,78.9)
        } ) ;
        stu3.setStudentCourses(new StudentCourse[] {
                new StudentCourse(stu3,cb,98.9)
        } ) ;
        // 3、设置课程和学生关系
        ca.setStudentCourses(new StudentCourse[] {
                new StudentCourse(stu1,ca,99.9),
                new StudentCourse(stu2,ca,78.9)
        } ) ;
        cb.setStudentCourses(new StudentCourse[] {
                new StudentCourse(stu1,cb,87.0),
                new StudentCourse(stu3,cb,98.9)
        } ) ;
        // 第二步：根据结构取出数据
        // 1、可以找到一门课程，以及参加此课程的所有的学生信息，和他的成绩；
        System.out.println(ca.getInfo()) ;
        for (int x = 0 ; x < ca.getStudentCourses().length ; x ++) {
                System.out.print("\t|- " + ca.getStudentCourses()[x].getStudent().getInfo()) ;
                System.out.print("，成绩：" + ca.getStudentCourses()[x].getScore()) ;
                System.out.println() ;
        }
        System.out.println("======================================") ;
        System.out.println(stu1.getInfo()) ;
        for (int x = 0 ; x < stu1.getStudentCourses().length;x ++) {
                System.out.print("\t|- " + stu1.getStudentCourses()[x].getCourse().getInfo()) ;
                System.out.print("，成绩：" + stu1.getStudentCourses()[x].getScore()) ;
                System.out.println() ;
        }
    }
}
```

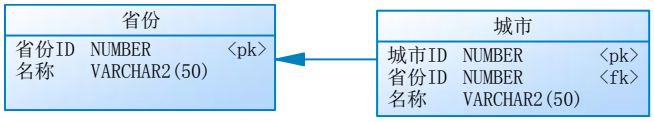
这些关系的开发模式必须灵活编写，随便转换。

3、总结

实体关系转换 —— 简单 Java 类进阶。

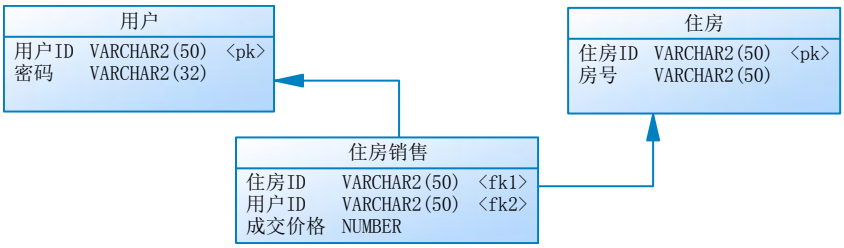
4、作业

1、 一对多关系：



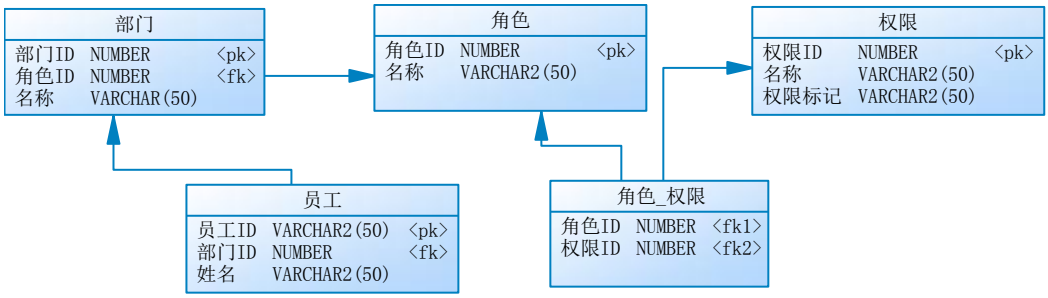
要求可以输出一个省份的信息，以及此省份下的所有城市信息；

2、 多对多



可以根据一个用户查找所有其对应的住房信息以及价格；

3、 多对多



- 要求可以根据一个员工找到他所对应的部门，以及该部门对应的角色，以及每个角色对应的所有权限；
- 可以根据一个角色找到具备此角色的所有部门，以及该部门下的所有员工；
- 根据一个权限列出具备有该权限的所有的角色以及每一个角色下对应的所有部门，以及每个部门中的所有员工。