Mybatis的执行连接查询(mybatis关联查询)

课程目标

1. Mybatis执行连接查询(多对一)

业务：查询所有学生及对应的年级信息

1. (重点理解)类与类之间的关系
2. (重点)使关联映射实现从属查询

(一对多查询)

业务:显示每个年级及年级下的学生

-- 回顾

1.常用关系型数据库:mysql\oracle\SQL SERVER

2.如何实现表与表之间的关系

表与表之间的关系是通过主外键(公共字段)形成.

3.表与表之间的关系分类

1对1、 1对多、多对一、多对多

1. 实现查询所有学生及对应的年级名称:-🡪连接查询

回顾连接查询

|  |
| --- |
| -- 回顾连接查询的语法  1.连接查询分类  INNER JOIN 指内连接、left JOIN 左外连接、right JOIN 右外连接  2.连接查询语法  SELECT \* FROM 表1 INNER JOIN 表2 ON 表1.公共字段=表2.公共字段 |

1. 修改学生实体类添加年级名称属性

Private String gname;

Setter和getter方略

1. 编写dao层

接口:

|  |
| --- |
| *//查询所有学生及对应的年级名称* List<Student> getAllStudentInfo(); |

持久化操作:

|  |
| --- |
| *<!--查询学生及关联的年级名称-->* <**select id="getAllStudentInfo" resultType="Student"**>  SELECT student.*\**,gname FROM student LEFT JOIN grade  ON student.gid=grade.`gid` </**select**> |

1. 测试

|  |
| --- |
| **public static void** main(String[] args) {  *//测试dao  //1.获取dao层对象* SqlSession sqlSession=MyBatisUtil.*getSession*();  StudentsMapper studentsMapper=sqlSession.getMapper(StudentsMapper.**class**);  *//2.调用* List<Students> list=studentsMapper.selectStudentInfo();  *//3.显示数据* System.***out***.println(**"学号\t姓名\t年龄\t性别\t生日\t年级名称"**);  list.forEach(s -> {  System.***out***.println(s.getXh()+**"\t"**+s.getName()+**"\t"**+s.getAge()+**"\t"**+s.getSex()+**"\t"**+s.getBirthday()+**"\t"**+s.getGname());  }); } |

1. 理解：类与类之间的关系
2. 表与表之间的关系

1.1 构成关系:

表与表之间是通过主外键(公共字段)形成的关系

* 1. 关系的分类

1对1、1对多、多对1、多对多

1. 类与类之间的关系

2.1 利用类的属性实现类与类之间的关系

人类Person

|  |
| --- |
| public class Person{  //定义车对象  Car car=new Car(); //1对1关系  //定义车的集合对象  List<Car> cars=new ArrayList<Car>(); //1对多关系  } |

车类Car

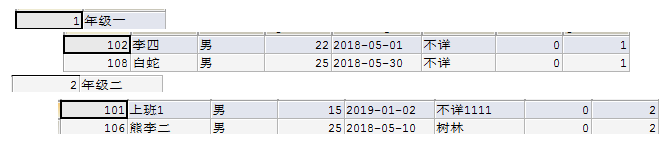
|  |
| --- |
| Public class Car{  //定义人对象  Person person=new Person(); //1对1 或者 多对1  //坐多个  List<Person> plist=new ArrayList<Person>(); //一对多关系  } |

2.2 类与类之间的关系:1对1 、1对多、多对1、多对多

1. 从属查询(一对多查询):查询年级信息及每个年级对应的学生信息

使用mybatis的关系映射

需求分析:



1. 修改年级实体类(一对多关系)：实现一个年级多名学生的关系

|  |
| --- |
| **public class** Grade {  **private** Short **gid**;  **private** String **gname**;  *//添加学生集合属性:实现一个年级多名学生* **private** List<Students> **students**=**new** ArrayList<>();  **public** List<Students> getStudents() {  **return students**;  }  **public void** setStudents(List<Students> students) {  **this**.**students** = students;  }  } |

1. 编写dao层

接口:

|  |
| --- |
| *//查询年级及对应的学生* List<Grade> getGradeBelongStu(); |

持久化操作:利用mybatis的关联映射实现一对多查询

|  |
| --- |
| *<!--查询所有年级及对应的学生  执行连接查询，利用resultMap返回复杂结果的映射  实现一对多的映射  -->* <**resultMap id="result1" type="Grade"**>  <**id column="gid" property="gid"**></**id**>  <**id column="gname" property="gname"**></**id**>  <!--使用collection实现一对多的关联映射  property 指定集合属性  javaType 指定是集合的类型  ofType 指定集合关联的泛型  notNullColumn="xh" 指定学号，如果学号为空时不返回对象，不为空时返回对象  *autoMapping="true" 实现属性和列的自动映射(字段名要与属性名相同)*  --><**collection property="students" javaType="list" ofType="Student" notNullColumn="xh"** *autoMapping="true"*>  <!--  <**id column="xh" jdbcType="INTEGER" property="xh"** />  <**result column="name" jdbcType="VARCHAR" property="name"** />  <**result column="age" jdbcType="TINYINT" property="age"** />  <**result column="sex" jdbcType="VARCHAR" property="sex"** />  <**result column="birthday" jdbcType="DATE" property="birthday"** />  <**result column="state" jdbcType="TINYINT" property="state"** />  <**result column="address" jdbcType="VARCHAR" property="address"** />  <**result column="gid" jdbcType="INTEGER" property="gid"** />  -->   </**collection**>  </**resultMap**>  <**select id="getGradeBelongStu" resultMap="result1"**>  SELECT *\** FROM grade LEFT JOIN student ON grade.`gid`=student.gid  </**select**> |

测试dao的代码:

|  |
| --- |
| **public static void** main(String[] args) {  *//获取sqlSession* SqlSession sqlSession=MyBatisUtil.*getSession*();  *//创建dao层对象* GradeMapper gradeMapper=sqlSession.getMapper(GradeMapper.**class**);  List<Grade> list= gradeMapper.getGradeBelongStu();  *//显示* System.***out***.println(**"编号\t名称"**);  list.forEach(grade -> {  System.***out***.println(grade.getGid()+**"\t"**+grade.getGname());  System.***out***.println(**"显示该年级的下的学生...."**);  *//获取当前年级的学生* List<Student> students=grade.getStudents();  students.forEach(s -> {  System.***out***.println(**"====学号:"**+s.getXh()+**"=姓名:"**+s.getName()+**"=年龄:"**+s.getAge()+**"=性别:"**+s.getSex());  });  });  sqlSession.close(); } |

1. resultType和resultMap的区别

注意: resultType和resultMap不能共存

1. resultMap用于返回复杂结果(比如:1对多关联射)类型:手动实现字段与属性映射

理解为自定义返回结果

1. resultType返回非关联映射的结果
2. (不推荐 )实现查询所有学生及对应的年级名称.(多对一)

利用mybatis的关联映射

1. 修改学生的实体类，添加年级对象属性(实现一个学生对应一个年级)

|  |
| --- |
| **public class** Student {   **学生相关的属性….**  //关联年级对象   **private** Grade **grade**;   **public** Grade getGrade() {  **return grade**;  }  **public void** setGrade(Grade grade) {  **this**.**grade** = grade;  }  } |

1. 编写dao层方法查询所有学生及对应的年级

|  |
| --- |
| *//查询年级及对应的学生* List<Student> getAllStudentInfo2(); |

1. 编写持久化操作 利用resultMap实现多对一关联映射

|  |
| --- |
| *<!--查询学生关联的年级 多对一的映射-->* <**resultMap id="result1" type="Student"**>  <**id column="xh" jdbcType="INTEGER" property="xh"** />  <**result column="name" jdbcType="VARCHAR" property="name"** />  <**result column="age" jdbcType="TINYINT" property="age"** />  <**result column="sex" jdbcType="VARCHAR" property="sex"** />  <**result column="birthday" jdbcType="DATE" property="birthday"** />  <**result column="state" jdbcType="TINYINT" property="state"** />  <**result column="address" jdbcType="VARCHAR" property="address"** />  <**result column="gid" jdbcType="INTEGER" property="gid"** />  *<!--使用association实现多（一）对一映射-->* <**association property="grade" javaType="Grade"**>  <**result column="gname" jdbcType="VARCHAR" property="gname"** />  </**association**> </**resultMap**> <**select id="getAllStudentInfo2" resultMap="result1"**>  SELECT *\** FROM student LEFT JOIN grade  ON student.gid=grade.`gid` </**select**> |

1. 编写主函数测试查询

|  |
| --- |
| **public static void** main(String[] args) {  *//调用dao层执行持久化操作得到结果* SqlSession sqlSession= MyBatisUtil.*getSession*();  *//创建dao层对象* StudentMapper studentMapper=sqlSession.getMapper(StudentMapper.**class**);  List<Student> list=studentMapper.getAllStudentInfo2();  *//输出信息* System.***out***.println(**"学号\t姓名\t年龄\t性别\t生日\t年级名称"**);  list.forEach(s -> {  System.***out***.println(s.getXh()+**"\t"**+s.getName()+**"\t"**+s.getAge()+**"\t"**+s.getSex()+**"\t"**+s.getBirthday()+**"\t"**+s.getGrade().getGname());  });   MyBatisUtil.*closeSession*(); } |