多对多查询

二、示例：用户和角色

一个用户可以有多个角色

一个角色可以赋予多个用户

步骤：

1、建立两张表：用户表，角色表

让用户表和角色表具有多对多的关系。需要使用中间表，中间表中包含各自的主键，在中间表中是外键。

2、建立两个实体类：用户实体类和角色实体类

让用户和角色的实体类能体现出来多对多的关系

各自包含对方一个集合引用

3、建立两个配置文件

用户的配置文件

角色的配置文件

4、实现配置：

当我们查询用户时，可以同时得到用户所包含的角色信息

当我们查询角色时，可以同时得到角色的所赋予的用户信息

实现 Role 到 User 多对多

1、编写User 类（省略构造get,set）

```
public class User implements Serializable {
    private Integer id;
    private String username;
    private String address;
    private String sex;
    private Date birthday;

    //多对多的关系映射：一个用户可以具备多个角色
    private List<Role> roles;

    public List<Role> getRoles() {
        return roles;
    }
}
```

2、编写Role类（省略构造get,set）

```
public class Role implements Serializable {

    private Integer roleId;
    private String roleName;
    private String roleDesc;

    //多对多的关系映射：一个角色可以赋予多个用户
    private List<User> users;

    public List<User> getUsers() {
        return users;
    }
}
```

3、编写 Role 持久层接口

```
public interface IRoleDao {
/* * 查询所有角色
List<Role> findAll();
}
```

4、编写映射配置文件IRole.xml

注意：在插入sql语句换行时要添加空格

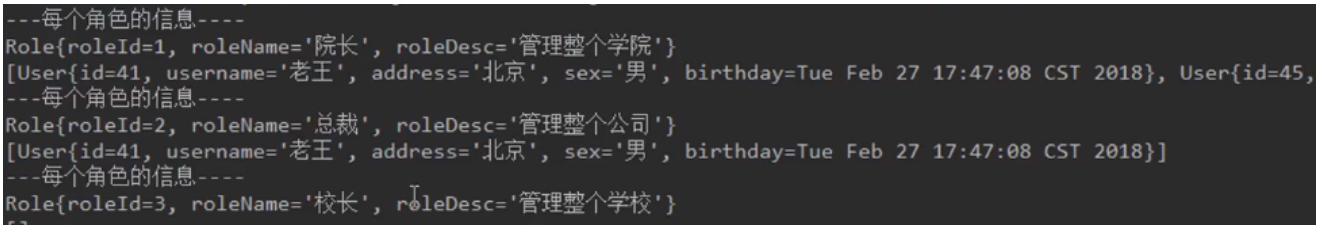
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
      PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
      "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.itheima.dao.IRoleDao">

   <!--定义role表的ResultMap-->
   <resultMap id="roleMap" type="role">
      <id property="roleId" column="rid"></id>
      <result property="roleName" column="role_name"></result>
      <result property="roleDesc" column="role_desc"></result>
      <collection property="users" ofType="user">
         <id column="id" property="id"></id>
         <result column="username" property="username"></result>
         <result column="address" property="address"></result>
         <result column="sex" property="sex"></result>
         <result column="birthday" property="birthday"></result>
      </collection>
   </resultMap>

   <!--查询所有-->
   <select id="findAll" resultMap="roleMap">
     select u.*,r.id as rid,r.role_name,r.role_desc from role r
      left outer join user_role ur  on r.id = ur.rid
      left outer join user u on u.id = ur.uid
   </select>
</mapper>
```

5、测试

```
public class RoleTest {
   private InputStream in;
   private SqlSession sqlSession;
   private IRoleDao roleDao;

   @Before//用于在测试方法执行之前执行
   public void init()throws Exception{
      //1.读取配置文件，生成字节输入流
      in = Resources.getResourceAsStream("SqlMapConfig.xml");
      //2.获取SqlSessionFactory
      SqlSessionFactory factory = new SqlSessionFactoryBuilder().build(in);
      //3.获取SqlSession对象
      sqlSession = factory.openSession(true);
      //4.获取dao的代理对象
      roleDao = sqlSession.getMapper(IRoleDao.class);
   }

   @After//用于在测试方法执行之后执行
   public void destroy()throws Exception{
      //提交事务
      // sqlSession.commit();
      //6.释放资源
      sqlSession.close();
      in.close();
   }

   /**
```

```
    * 测试查询所有
    */
    @Test
    public void testFindAll(){
        List<Role> roles = roleDao.findAll();
        for(Role role : roles){
            System.out.println("---每个角色的信息----");
            System.out.println(role);
            System.out.println(role.getUsers());
        }
    }
    }
```

6、测试结果



实现 User 到 Role 多对多

1、在上面的基础上，编写 User 持久层接口

```
public interface IUserDao {
    /**
     * 查询所有用户，同时获取到用户下所有账户的信息
     */
    List<User> findAll();

    /**
     * 根据id查询用户信息
     * @param userId
     */
    User findById(Integer userId);
}
```

2、编写映射配置文件IUser.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.itheima.dao.IUserDao">

    <!-- 定义User的resultMap-->
    <resultMap id="userMap" type="user">
        <id property="id" column="id"></id>
        <result property="username" column="username"></result>
        <result property="address" column="address"></result>
        <result property="sex" column="sex"></result>
        <result property="birthday" column="birthday"></result>
        <!-- 配置角色集合的映射 -->
        <collection property="roles" ofType="role">
            <id property="roleId" column="rid"></id>
            <result property="roleName" column="role_name"></result>
            <result property="roleDesc" column="role_desc"></result>
        </collection>
    </resultMap>

    <!-- 查询所有 -->
    <select id="findAll" resultMap="userMap">
```

```
        select u.*,r.id as rid,r.role_name,r.role_desc from user u
         left outer join user_role ur  on u.id = ur.uid
         left outer join role r on r.id = ur.rid
    </select>

    <!-- 根据id查询用户 -->
    <select id="findById" parameterType="INT" resultType="user">
        select * from user where id = #{uid}
    </select>
</mapper>
```

3、测试

```
public class UserTest {
    private InputStream in;
    private SqlSession sqlSession;
    private IUserDao userDao;

    @Before//用于在测试方法执行之前执行
    public void init()throws Exception{
        //1.读取配置文件，生成字节输入流
        in = Resources.getResourceAsStream("SqlMapConfig.xml");
        //2.获取SqlSessionFactory
        SqlSessionFactory factory = new SqlSessionFactoryBuilder().build(in);
        //3.获取SqlSession对象
        sqlSession = factory.openSession(true);
        //4.获取dao的代理对象
        userDao = sqlSession.getMapper(IUserDao.class);
    }

    @After//用于在测试方法执行之后执行
    public void destroy()throws Exception{
        //提交事务
        // sqlSession.commit();
        //6.释放资源
        sqlSession.close();
        in.close();
    }

    /**
     * 测试查询所有
     */
    @Test
    public void testFindAll(){
        List<User> users = userDao.findAll();
        for(User user : users){
            System.out.println("-----每个用户的信息------");
            System.out.println(user);
            System.out.println(user.getRoles());
        }
    }
}
```

4、测试结果

```
-----每个用户的信息------
User{id=41, username='老王', address='北京', sex='男', birthday=Tue Feb 27 17:47:08 CST 2018}
[Role{roleId=1, roleName='院长', roleDesc='管理整个学院'}, Role{roleId=2, roleName='总裁', roleDesc='管理整个公
-----每个用户的信息------
User{id=42, username='小二王', address='北京金燕龙', sex='女', birthday=Fri Mar 02 15:09:37 CST 2018}
[]
-----每个用户的信息------
User{id=43, username='小二王', address='北京金燕龙', sex='女', birthday=Sun Mar 04 11:34:34 CST 2018}
[]
-----每个用户的信息------
User{id=45, username='传智播客', address='北京金燕龙', sex='男', birthday=Sun Mar 04 12:04:06 CST 2018}
[Role{roleId=1, roleName='院长', roleDesc='管理整个学院'}]
-----每个用户的信息------
User{id=46, username='老王', address='北京', sex='女', birthday=Wed Mar 07 17:37:26 CST 2018}
[]
-----每个用户的信息------
User{id=50, username='userdaoimpl update user', address='北京市顺义区', sex='女', birthday=Tue Jun 12 01:13:0
```

对应数据库：

id	username	birthday	sex	address	rid	role_name	role_desc
41	老王	2018-02-27 17:4	男	北京	1	院长	管理整个学院
41	老王	2018-02-27 17:4	男	北京	2	总裁	管理整个公司
42	小二王	2018-03-02 15:0	女	北京金燕龙	(Null)	(Null)	(Null)
43	小二王	2018-03-04 11:3	女	北京金燕龙	(Null)	(Null)	(Null)
45	传智播客	2018-03-04 12:0	男	北京金燕龙	1	院长	管理整个学院
46	老王	2018-03-07 17:3	女	北京	(Null)	(Null)	(Null)
50	userdaoimpl up	2018-06-12 01:1	女	北京市顺义区	(Null)	(Null)	(Null)
51	testid	2018-01-01 00:0	男	bj	(Null)	(Null)	(Null)
52	mybatis last ins	2018-06-12 00:0	男	北京市顺义区	(Null)	(Null)	(Null)
53	modify User pro	2018-06-12 00:2	男	北京市顺义区	(Null)	(Null)	(Null)
55	dao impl user	2018-06-12 01:1	男	北京市顺义区	(Null)	(Null)	(Null)
56	autocommit	2018-06-12 17:2	男	北京市顺义区	(Null)	(Null)	(Null)