**关联查询**

# 商品订单数据模型

用户表：user

记录了购买商品的用户信息

Id：唯一标识一个用户

订单表：orders

记录了用户创建的订单

创建用户:user\_id(外键)

订单号

创建时间

订单状态

外键：

user\_id

订单明细表：orderdetail

记录了用户购买信息

所属订单：orders\_id（外键）

商品id:items\_id（外键）

商品数量

商品购买价格

外键：orders\_id

商品信息：items

记录了所有商品信息

商品id：id（主键）

商品名称：

商品介绍

商品价格

外键：

items\_id

一对多

一个用户可以创建多个订单

一对一：

一个订单只能由一个用户创建

一对多

一个订单包括多个购买明细

一对一

一个订单明细只能

属于一个订单

一对一

一个明细对应一个商品

一对多

一个商品对应多个订单明细

通过订单明细表订单表和商品表建立 关系

一个订单对应多个商品

一个商品对应多个订单

订单表和商品表是多对多关系

用户与订单是一对多

订单和商品是多对多

用户和商品是多对多

# 一对一查询

案例：查询所有订单信息，关联查询下单用户信息。

注意：因为一个订单信息只会是一个人下的订单，所以从查询订单信息出发关联查询用户信息为一对一查询。如果从用户信息出发查询用户下的订单信息则为一对多查询，因为一个用户可以下多个订单。

## 方法一(了解)：

使用resultType，定义订单信息po类，此po类中包括了订单信息和用户信息：

### Sql语句：

SELECT

orders.\*,

user.username,

user.address

FROM

orders,

user

WHERE orders.user\_id = user.id

### 定义po类

Po类中应该包括上边sql查询出来的所有字段，如下：

**public** **class** OrdersCustom **extends** Orders {

**private** String username;// 用户名称

**private** String address;// 用户地址

get/set。。。。

OrdersCustom类继承Orders类后OrdersCustom类包括了Orders类的所有字段，只需要定义用户的信息字段即可。

### Mapper.xml

<!-- 查询所有订单信息 -->

<select id=*"findOrdersList"* resultType=*"com.mybatis.po.OrdersCustom"*>

SELECT

orders.\*,

user.username,

user.address

FROM

orders, user

WHERE orders.user\_id = user.id

</select>

### Mapper接口：

public List<*OrdersCustom*> findOrdersList() throws Exception;

### 测试：

**Public void** testfindOrdersList()**throws** Exception{

//获取session

SqlSession session = sqlSessionFactory.openSession();

//获限mapper接口实例

UserMapper userMapper = session.getMapper(UserMapper.**class**);

//查询订单信息

List<*OrdersCustom*> list = userMapper.findOrdersList();

System.*out*.println(list);

//关闭session

session.close();

}

### 总结：

定义专门的po类作为输出类型，其中定义了sql查询结果集所有的字段。此方法较为简单，企业中使用普遍。

## 方法二(重点)：

使用resultMap，定义专门的resultMap用于映射一对一查询结果。

### Sql语句：

SELECT

orders.\*,

user.username,

user.address

FROM

orders,

user

WHERE orders.user\_id = user.id

### 定义po类

在Orders类中加入User属性，user属性中用于存储关联查询的用户信息，因为订单关联查询用户是一对一关系，所以这里使用单个User对象存储关联查询的用户信息。

### Mapper.xml

<select id=*"findOrdersListResultMap"* resultMap=*"userordermap"*>

SELECT

orders.\*,

user.username,

user.address

FROM

orders, user

WHERE orders.user\_id = user.id

</select>

这里resultMap指定*userordermap。*

### 定义resultMap

需要关联查询映射的是用户信息，使用association将用户信息映射到订单对象的用户属性中。

<resultMap type="com.mybatis.po.Orders" id="userordermap">

<!-- 这里的id是订单对象属性与订单表字段映射 -->

<id property="id" column="id" />

<result property="number" column="number" />

<result property="createTime" column="createTime" />

<result property="note" column="note" />

<association property="user" javaType="com.mybatis.po.User">

<!-- 这里的id为user的id，如果写上表示给user的id属性赋值 -->

<id property="id" column="user\_id" />

<result property="username" column="username" />

<result property="address" column="address" />

</association>

</resultMap>

association：表示进行关联查询单条记录

property：表示关联查询的结果存储在com.mybatis.po.Orders的user属性中

javaType：表示关联查询的结果类型

<id property=*"id"* column=*"user\_id"*/>：查询结果的user\_id列对应关联对象的id属性，这里是<id />表示user\_id是关联查询对象的唯一标识。

<result property=*"username"* column=*"username"*/>：查询结果的username列对应关联对象的username属性。

### Mapper接口：

public List<Orders> findOrdersListResultMap() throws Exception;

### 测试：

// 创建数据库会话实例sqlSession

session = sqlSessionFactory.openSession();

// 查询单个记录，根据用户id查询用户信息

OrdersMapper ordersMapper = session.getMapper(OrdersMapper.**class**);

List<Orders> userList = ordersMapper.findOrdersListResultMap();

**for** (Orders o : userList) {

System.***out***.println(o.getUser().getId()+"\t"+o.getId()+"\t"+o.getNote()+"\t"+o.getNumber()+"\t"+o.getCreateTime()+"\t"+o.getUser().getUsername());

}

System.***out***.println(userList);

### 小结：

使用association完成关联查询，将关联查询信息映射到pojo对象中。

# 一对多查询

案例：查询所有订单信息及订单下的订单明细信息。

订单信息与订单明细为一对多关系。

使用resultMap实现如下：

## Sql语句：

SELECT

orders.\*,

user.username,

user.address,

orderdetail.id orderdetail\_id,

orderdetail.items\_id,

orderdetail.items\_num

FROM

orders,user,orderdetail

WHERE orders.user\_id = user.id

AND orders.id = orderdetail.orders\_id

## 定义po类

在Orders类中加入User属性。

在Orders类中加入List<Orderdetail> orderdetails属性

## Mapper.xml

<select id=*"findOrdersDetailList"* resultMap=*"userorderdetailmap"*>

SELECT

orders.\*,

user.username,

user.address,

orderdetail.id orderdetail\_id,

orderdetail.items\_id,

orderdetail.items\_num

FROM orders,user,orderdetail

WHERE orders.user\_id = user.id

AND orders.id = orderdetail.orders\_id

</select>

## 定义resultMap

<!-- 订单信息resultmap -->

<resultMap type=*"com.mybatis.po.Orders"* id=*"userorderdetailmap"*>

<id property=*"id"*column=*"id"*/>

<result property=*"user\_id"* column=*"user\_id"*/>

<result property=*"number"* column=*"number"*/>

<association property=*"user"* javaType=*"com.mybatis.po.User"*>

<id property=*"id"* column=*"user\_id"*/>

<result property=*"username"* column=*"username"*/>

<result property=*"address"* column=*"address"*/>

</association>

<collection property=*"orderdetails"* ofType=*"com.mybatis.po.Orderdetail"*>

<id property=*"id"* column=*"orderdetail\_id"*/>

<result property=*"items\_id"* column=*"items\_id"*/>

<result property=*"items\_num"* column=*"items\_num"*/>

</collection>

</resultMap>

黄色部分和上边一对一查询订单及用户信息定义的resultMap相同，

collection部分定义了查询订单明细信息。

collection：表示关联查询结果集

property=*"orderdetails"：*关联查询的结果集存储在com.mybatis.po.Orders上哪个属性。

ofType=*"com.mybatis.po.Orderdetail"：*指定关联查询的结果集中的对象类型即List中的对象类型。

<id />及<result/>的意义同一对一查询。

### resultMap使用继承

上边定义的resultMap中黄色部分和一对一查询订单信息的resultMap相同，这里使用继承可以不再填写重复的内容，如下：

<resultMap type=*"com.mybatis.po.Orders"* id=*"userorderdetailmap"* extends=*"userordermap"*>

<collection property=*"orderdetails"* ofType=*"com.mybatis.po.Orderdetail"*>

<id property=*"id"* column=*"orderdetail\_id"*/>

<result property=*"items\_id"* column=*"items\_id"*/>

<result property=*"items\_num"* column=*"items\_num"*/>

</collection>

</resultMap>

使用extends继承订单信息*userordermap。*

## Mapper接口：

public List<Orders>findOrdersDetailList () throws Exception;

## 测试：

**public** **void** testfindOrdersDetailList()**throws** Exception{

//获取session

SqlSession session = sqlSessionFactory.openSession();

//获限mapper接口实例

OrdersMapper userMapper = session.getMapper(OrdersMapper.**class**);

//查询订单信息

List<Orders> list = userMapper.findOrdersDetailList();

**for** (Orders o : list) {

**for** (Orderdetail od : o.getOrderdetails()) {

System.***out***.println(od.getItemsId());

}

}

System.***out***.println(list);

//关闭session

session.close();

}

## 小结

如果两个表都有同名的id字段，则必须给其中一个取别名区分开，否则查的记录数条数会有问题,在映射文件中使用“别名”进行映射

# 多对多查询

## 查询用户购买的商品信息

### 需求

查询用户购买的商品信息。

### sql

需要查询所有用户信息，关联查询订单及订单明细信息，订单明细信息中关联查询商品信息

SELECT

orders.\*,

USER .username,

USER .address,

orderdetail.id orderdetail\_id,

orderdetail.items\_id,

orderdetail.items\_num,

items.name items\_name,

items.detail items\_detail

FROM

orders,

USER,

orderdetail,

items

WHERE

orders.user\_id = USER .id

AND orders.id = orderdetail.orders\_id

AND orderdetail.items\_id = items.id

### po定义

在*User*中添加List<Orders> orders 属性，在Orders类中加入List<Orderdetail> orderdetails属性,在Orderdetail添加Items属性

### resultMap

需要关联查询映射的信息是：订单、订单明细、商品信息

订单：一个用户对应多个订单，使用collection映射到用户对象的订单列表属性中

订单明细：一个订单对应多个明细，使用collection映射到订单对象中的明细属性中

商品信息：一个订单明细对应一个商品，使用association映射到订单明细对象的商品属性中。

<!-- 一对多查询

查询用户信息、关联查询订单、订单明细信息、商品信息

-->

<resultMap type=*"com.mybatis.po.User"* id=*"userOrderListResultMap"*>

<id column=*"user\_id"* property=*"id"*/>

<result column=*"username"* property=*"username"*/>

<collection property=*"orders"* ofType=*"com.mybatis.po.Orders"*>

<id column=*"id"* property=*"id"*/>

<result property=*"number"* column=*"number"*/>

<collection property=*"orderdetails"* ofType=*"com.mybatis.po.Orderdetail"*>

<id column=*"orderdetail\_id"* property=*"id"*/>

<result property=*"ordersId"* column=*"id"*/>

<result property=*"itemsId"* column=*"items\_id"*/>

<result property=*"itemsNum"* column=*"items\_num"*/>

<association property=*"items"* javaType=*"com.mybatis.po.Items"*>

<id column=*"items\_id"* property=*"id"*/>

<result column=*"items\_name"* property=*"name"*/>

<result column=*"items\_detail"* property=*"detail"*/>

</association>

</collection>

</collection>

</resultMap>

## 小结

一对多是多对多的特例，如下需求：

查询用户购买的商品信息，用户和商品的关系是多对多关系。

需求1：

查询字段：用户账号、用户名称、用户性别、商品名称、商品价格(最常见)

企业开发中常见明细列表，用户购买商品明细列表，

使用resultType将上边查询列映射到pojo输出。

需求2：

查询字段：用户账号、用户名称、购买商品数量、商品明细（鼠标移上显示明细）

使用resultMap将用户购买的商品明细列表映射到user对象中。

# resultMap小结

resultType：

作用：

将查询结果按照sql列名pojo属性名一致性映射到pojo中。

场合：

常见一些明细记录的展示，比如用户购买商品明细，将关联查询信息全部展示在页面时，此时可直接使用resultType将每一条记录映射到pojo中，在前端页面遍历list（list中是pojo）即可。

resultMap：

使用association和collection完成一对一和一对多高级映射（对结果有特殊的映射要求）。

association：

作用：

将关联查询信息映射到一个pojo对象中。

场合：

为了方便查询关联信息可以使用association将关联订单信息映射为用户对象的pojo属性中，比如：查询订单及关联用户信息。

使用resultType无法将查询结果映射到pojo对象的pojo属性中，根据对结果集查询遍历的需要选择使用resultType还是resultMap。

collection：

作用：

将关联查询信息映射到一个list集合中。

场合：

为了方便查询遍历关联信息可以使用collection将关联信息映射到list集合中，比如：查询用户权限范围模块及模块下的菜单，可使用collection将模块映射到模块list中，将菜单列表映射到模块对象的菜单list属性中，这样的作的目的也是方便对查询结果集进行遍历查询。

如果使用resultType无法将查询结果映射到list集合中。

# 延迟加载

需要查询关联信息时，使用mybatis延迟加载特性可有效的减少数据库压力，首次查询只查询主要信息，关联信息等用户获取时再加载。

## 打开延迟加载开关

在mybatis核心配置文件中配置：

lazyLoadingEnabled、aggressiveLazyLoading

|  |  |  |  |
| --- | --- | --- | --- |
| 设置项 | 描述 | 允许值 | 默认值 |
| lazyLoadingEnabled | 全局性设置懒加载。如果设为‘false’，则所有相关联的都会被初始化加载。 | true | false | false |
| aggressiveLazyLoading | 当设置为‘true’的时候，懒加载的对象可能被任何懒属性全部加载。否则，每个属性都按需加载。 | true | false | true |

<settings>

<setting name=*"lazyLoadingEnabled"* value=*"true"*/>

<setting name=*"aggressiveLazyLoading"* value=*"false"*/>

</settings>

## 一对一查询延迟加载

### 需求

查询订单信息，关联查询用户信息。

默认只查询订单信息，当需要查询用户信息时再去查询用户信息。

### Sql语句：

SELECT

orders.\*

FROM

orders

### 定义po类

在Orders类中加入User属性。

### Mapper.xml

<select id=*"findOrdersList3"* resultMap=*"userordermap2"*>

SELECT

orders.\*

FROM

orders

</select>

### 定义resultMap

<!-- 订单信息resultmap -->

<resultMap type=*"com.mybatis.po.Orders"* id=*"userordermap2"*>

<id property=*"id"* column=*"id"*/>

<result property=*"user\_id"* column=*"user\_id"*/>

<result property=*"number"* column=*"number"*/>

<association property=*"user"* javaType=*"com.mybatis.po.User"* select=*"findUserById"* column=*"user\_id"*/>

</resultMap>

association：

select=*"findUserById"*：指定关联查询sql为*findUserById*

column=*"user\_id"*：关联查询时将users\_id列的值传入*findUserById*

最后将关联查询结果映射至*com.mybatis.po.User。*

### Mapper接口：

public List<Orders> findOrdersList3() throws Exception;

### 测试：

**Public void** testfindOrdersList3()**throws** Exception{

//获取session

SqlSession session = sqlSessionFactory.openSession();

//获限mapper接口实例

UserMapper userMapper = session.getMapper(UserMapper.**class**);

//查询订单信息

List<Orders> list = userMapper.findOrdersList3();

System.out.println(list);

//开始加载，通过orders.getUser方法进行加载

for(Orders orders:list){

System.out.println(orders.getUser());

}

//关闭session

session.close();

}

### 延迟加载的思考

不使用mybatis提供的延迟加载功能是否可以实现延迟加载？

实现方法：

针对订单和用户两个表定义两个mapper方法。

1、订单查询mapper方法

2、根据用户id查询用户信息mapper方法

默认使用订单查询mapper方法只查询订单信息。

当需要关联查询用户信息时再调用根据用户id查询用户信息mapper方法查询用户信息。

## 一对多延迟加载

一对多延迟加载的方法同一对一延迟加载，在collection标签中配置select内容。

本部分内容自学。

## 延迟加载小结

作用：

当需要查询关联信息时再去数据库查询，默认不去关联查询，提高数据库性能。

只有使用resultMap支持延迟加载设置。

场合：

当只有部分记录需要关联查询其它信息时，此时可按需延迟加载，需要关联查询时再向数据库发出sql，以提高数据库性能。

当全部需要关联查询信息时，此时不用延迟加载，直接将关联查询信息全部返回即可，可使用resultType或resultMap完成映射。