# 分析数据模型—包装类型



## 思路

1. 明确每张表存储的信息
2. 明确每张表中关键字段（主键、外键、非空）
3. 明确数据库中表与表之间的外键关系
4. 明确业务中表与表的关系（建立在具体的业务）

## 图形分析

一对多

订单表

id：订单ID

Number：订单号

User\_id：用户id（外键）

订单明细表

Orders\_id：订单ID（外键）

Items\_id：商品ID（外键）

外键

一对一

外键

一对多

一对一

外键

一对一

一对多

多对多

多对多

商品表

Id：商品ID

用户表

Id：用户ID

## 外键关系

**user和orders：**

user---->orders：一个用户可以创建多个订单，一对多

orders--->user：一个订单只由一个用户创建，一对一

**orders和orderdetail：**

orders-🡪orderdetail：一个订单可以包括 多个订单明细，因为一个订单可以购买多个商品，每个商品的购买信息在orderdetail记录，一对多关系

orderdetail--> orders：一个订单明细只能包括在一个订单中，一对一

**orderdetail和itesm：**

orderdetail---》itesms：一个订单明细只对应一个商品信息，一对一

items--> orderdetail:一个商品可以包括在多个订单明细 ，一对多

## 没有外键关系

**Orders和items：**

这两张表没有直接的外键关系，通过业务及数据库的间接关系分析出它们是多对多的关系。

Orders🡪 orderdetail –>items：一个订单可以有多个订单明细，一个订单明细对应一个商品，所以一个订单对应多个商品

Items-🡪orderdetail🡪orders：一个商品可以对应多个订单明细，一个订单明细对应一个订单，所以一个商品对应多个订单

**User和items：**

这两张表没有直接的外键关系，通过业务及数据库的间接关系分析出它们是多对多的关系。

User🡪orders🡪orderdetail🡪items：一个用户有多个订单，一个订单有多个订单明细、一个订单明细对应一个商品，所以一个用户对应多个商品

Items🡪orderdetail🡪orders🡪user：一个商品对应多个订单明细，一个订单明细对应一个订单，一个订单对应一个用户，所以一个商品对应多个用户

# resultMap(入门)

## POJO自定义类型和list<pojo>



# 一对一查询

## 需求

查询订单信息关联查询用户信息

## sql语句

查询语句：

先确定主查询表：订单信息表

再确定关联查询表：用户信息

通过orders关联查询用户使用user\_id一个外键，只能关联查询出一条用户记录就可以使用内连接

SELECT

orders.\*,

user.username,

user.sex

FROM

orders,

USER

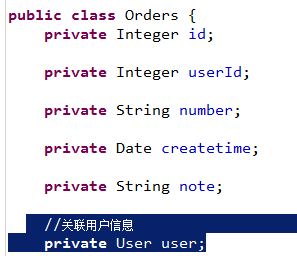
WHERE orders.user\_id = user.id

## 使用resultMap实现一对一

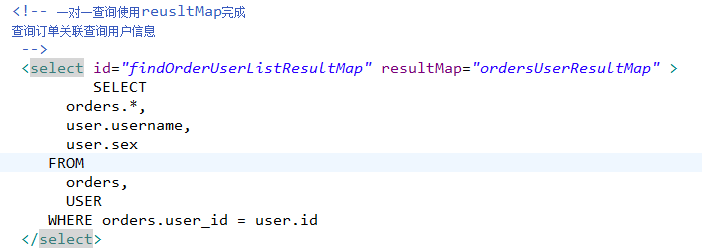
### resultMap映射思路

resultMap提供一对一关联查询的映射和一对多关联查询映射，一对一映射思路：将关联查询的信息映射到pojo中，如下：

在Orders类中创建一个User属性，将关联查询的信息映射到User属性中。



### mapper.xml



### resultMap定义

<!-- 一对一resultMap -->

<resultMap type=*"orders"* id=*"ordersUserResultMap"*>

<!-- 完成订单信息配置 -->

<!-- id :订单关联用户唯一标识 -->

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

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

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

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

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

<!--关联信息的映射 -->

<!--

association：用于对关联信息映射到单个pojo

property：要将关联信息映射到orders的哪个属性中

javaType：关联信息映射到orders的属性的类型，是user的类型

-->

<association property=*"user"* javaType=*"user"*>

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

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

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

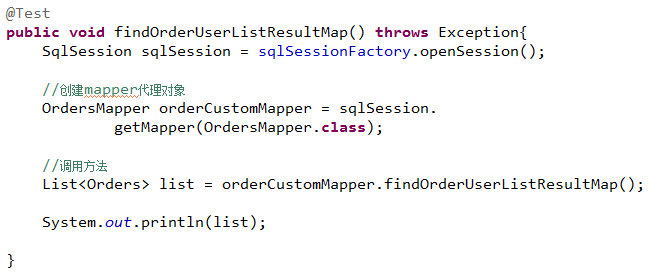
</association>

</resultMap>

### mapper.java



### 测试



## 小结

**resultType**：使用resultType实现较为简单，如果pojo中没有包括查询出来的列名，需要增加列名对应的属性，即可完成映射。

如果没有查询结果的特殊要求建议使用resultType。

**resultMap**：需要单独定义resultMap，实现有点麻烦，如果对查询结果有特殊的要求，使用resultMap可以完成将关联查询映射pojo的对象属性中。

**resultMap可以实现延迟加载，resultType无法实现延迟加载。**

# 一对多查询

## 需求

查询所有订单信息及订单下的订单明细信息。

## sql语句

主查询表：订单表

关联查询表：订单明细

SELECT

orders.\*,

user.username,

user.sex ,

orderdetail.id orderdetail\_id,

orderdetail.items\_num,

orderdetail.items\_id

FROM

orders,

USER,

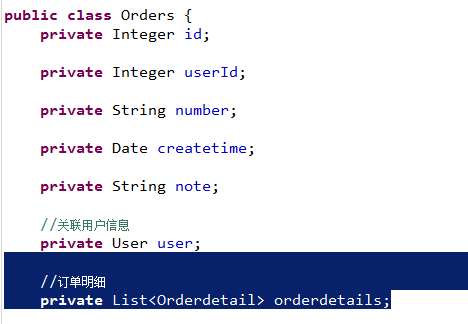
orderdetail

WHERE orders.user\_id = user.id AND orders.id = orderdetail.orders\_id

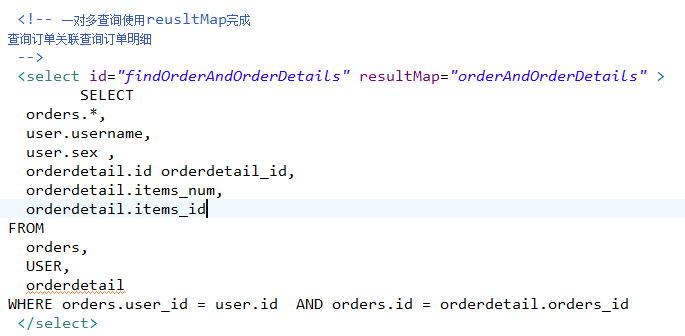
## pojo

resultMap 提供collection完成关联信息映射到集合对象中。

在orders类中创建集合属性：



## mapper.xml



## resultMap定义



## mapper.java



## 小结

mybatis使用resultMap的collection对关联查询的多条记录映射到一个list集合属性中。

**使用resultType实现：**

需要对结果集进行二次处理。

将订单明细映射到orders中的orderdetails中，需要自己处理，使用双重循环遍历，去掉重复记录，将订单明细放在orderdetails中。

# 一对多查询(复杂)

## 需求

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

## sql

主查询表：用户信息

关联查询：订单、订单明细，商品信息

SELECT

orders.\*,

user.username,

user.sex ,

orderdetail.id orderdetail\_id,

orderdetail.items\_num,

orderdetail.items\_id,

items.name items\_name,

items.detail items\_detail

FROM

orders,

USER,

orderdetail,

items

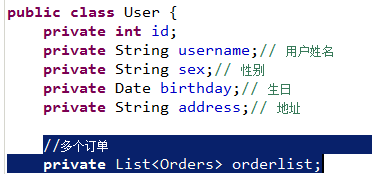
WHERE orders.user\_id = user.id AND orders.id = orderdetail.orders\_id AND items.id = orderdetail.items\_id

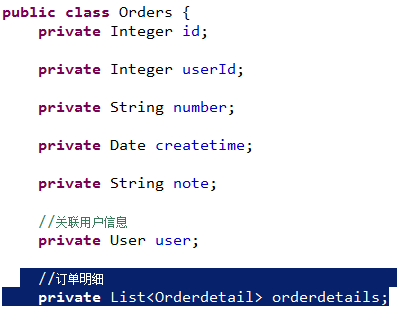
## pojo定义

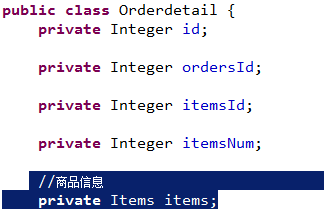
在user.java中创建映射的属性：集合 List<Orders> orderlist

在Orders中创建映射的属性：集合List<Orderdetail> orderdetails

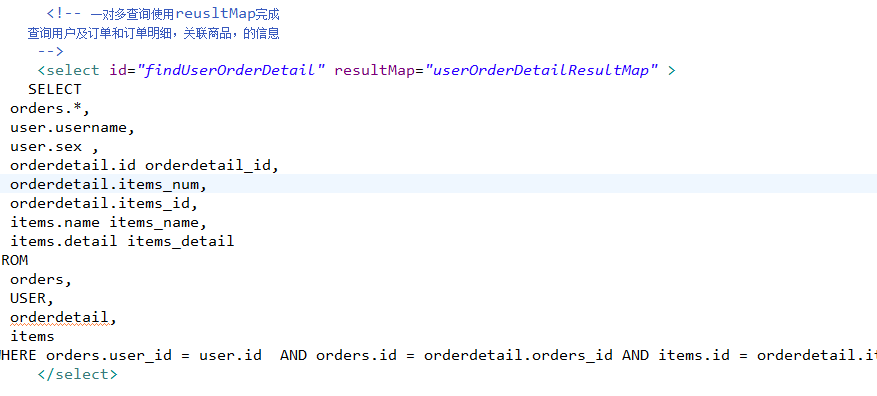
在Orderdetail中创建商品属性：pojo Items items







## mapper.xml



## resultMap



## mapper.java



# 多对多查询（自己实现 ）

## 需求

一对多是多对多的特例。

需求1：

查询显示字段：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：

查询显示字段：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。

示例二

## Sql

SELECT

user.id user\_id,

user.username user\_name,

orders.number orders\_number,

items.\*

FROM

orders,

USER,

orderdetail,

items

WHERE orders.user\_id = user.id

AND orders.id = orderdetail.orders\_id

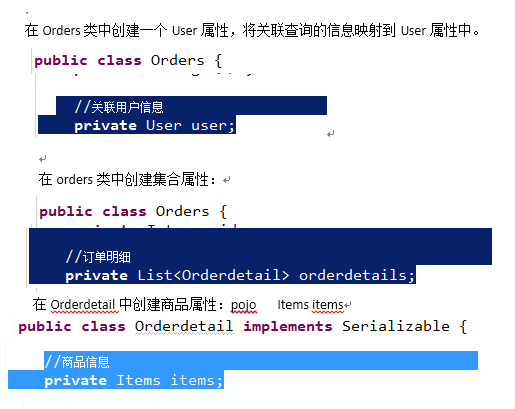
AND items.id = orderdetail.items\_id

## Pojo定义

在user.java中创建映射的属性：集合 List<Orders> orderlist

在Orders中创建映射的属性：集合List<Orderdetail> orderdetails

在Orderdetail中创建商品属性：pojo Items items



## Mapper.xml

<!-- 多对多 -->

<select id=*"findUserOrderItems"* resultMap=*"UserOrderItemsResultMap"*>

SELECT

user.id user\_id,

user.username user\_name,

orders.number orders\_number,

items.\*

FROM

orders,

USER,

orderdetail,

items

WHERE orders.user\_id = user.id

AND orders.id = orderdetail.orders\_id

AND items.id = orderdetail.items\_id

</select>

## reslutMap

<!-- 多对多 -->

<resultMap type=*"user"* id=*"UserOrderItemsResultMap"*>

<!-- user -->

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

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

<!-- order -->

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

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

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

<!-- Orderdetail -->

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

<id column=*"orders\_id"* property=*"ordersId"*/>

<id column=*"items\_Id"* property=*"itemsId"*/>

<!-- items -->

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

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

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

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

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

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

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

</association>

</collection>

</collection>

</resultMap>

## Mapper.java

@Test

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

SqlSession sqlSession = sqlSessionFactory.openSession();

//创建mapper代理对象

OrdersMapper ordersMapper = sqlSession.

getMapper(OrdersMapper.**class**);

//调用方法

List<User> list = ordersMapper.findUserOrderItems();

System.*out*.println(list);

}

## 小结

将查询用户购买的商品信息明细清单，（用户名、用户地址、购买商品名称、购买商品时间、购买商品数量）

针对上边的需求就使用resultType将查询到的记录映射到一个扩展的pojo中，很简单实现明细清单的功能。

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。

总结：

使用resultMap是针对那些对查询结果映射有特殊要求的功能，，比如特殊要求映射成list中包括 多个list。