内容

[使用联想实现延迟加载](https://github.com/brianway/springmvc-mybatis-learning/blob/master/mybatis/mybatis%E5%AD%A6%E4%B9%A0%E7%AC%94%E8%AE%B0(13)-%E5%BB%B6%E8%BF%9F%E5%8A%A0%E8%BD%BD.md#%E4%BD%BF%E7%94%A8association%E5%AE%9E%E7%8E%B0%E5%BB%B6%E8%BF%9F%E5%8A%A0%E8%BD%BD)

[延迟加载思考](https://github.com/brianway/springmvc-mybatis-learning/blob/master/mybatis/mybatis%E5%AD%A6%E4%B9%A0%E7%AC%94%E8%AE%B0(13)-%E5%BB%B6%E8%BF%9F%E5%8A%A0%E8%BD%BD.md#%E5%BB%B6%E8%BF%9F%E5%8A%A0%E8%BD%BD%E6%80%9D%E8%80%83)

[IDEA的调试小现象](https://github.com/brianway/springmvc-mybatis-learning/blob/master/mybatis/mybatis%E5%AD%A6%E4%B9%A0%E7%AC%94%E8%AE%B0(13)-%E5%BB%B6%E8%BF%9F%E5%8A%A0%E8%BD%BD.md#idea%E7%9A%84debug%E5%B0%8F%E7%8E%B0%E8%B1%A1)

resultMap可以实现高级映射（使用association，collection实现一对一及一对多映射）association，collection具备延迟加载功能。

延迟加载：先从单表查询，需要时再从关联表去关联查询，大大提高数据库性能，因为查询单表要比关联查询多张表速度要快。

需求：

如果查询订单并且关联查询用户信息。如果先查询订单信息即可满足要求，当我们需要查询用户信息时再查询用户信息。把对用户信息的按需去查询就是延迟加载。

使用联想实现延迟加载

mapper.xml

需要定义两个映射的方法对应的声明。

1.只查询订单信息

SELECT \* FROM orders

在查询订单的声明中使用联想去延迟加载（执行）下边的satatement（关联查询用户信息）

<！ -查询订单关联查询用户，用户信息需要延迟加载- >
< select id = “ findOrdersUserLazyLoading ” resultMap = “ OrdersUserLazyLoadingResultMap ” >
SELECT \* FROM订单
</ select >

2.关联查询用户信息

通过上边查询到的订单信息中的user\_id去关联查询用户信息，使用UserMapper.xml中的findUserById

< select id = “ findUserById ” parameterType = “ int ” resultType = “ com.iot.mybatis.po.User ” >
SELECT \* FROM user WHERE id =＃{value}
</ select >

上边先去执行findOrdersUserLazyLoading，当需要去查询用户的时候再去执行findUserById，通过结果映射的定义将延迟加载执行配置起来。

延迟加载的resultMap

<！ -延迟加载的resultMap - >
< resultMap type = “ com.iot.mybatis.po.Orders ” id = “ OrdersUserLazyLoadingResultMap ” >
<！ -对订单信息进行映射配置 - >
< 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 ” />
<！ -实现对用户信息进行延迟加载
select：指定延迟加载需要执行的语句的id（是根据user\_id查询用户信息的语句）
要使用userMapper。 xml中findUserById完成根据用户id（user\_id）用户信息的查询，如果findUserById不在本映射器中需要前边加命名空间
列：订单信息中关联用户信息查询的列，是user\_id
关联查询的sql理解为：
SELECT orders。\* ，
（SELECT用户名FROM USER WHERE orders.user\_id = user.id）用户名，
（SELECT sex FROM USER WHERE orders.user\_id = user.id）sex
FROM orders
- >
< association property = “ user “ javaType = ” com.iot.mybatis.po.User “
select = ” com.iot.mybatis.mapper.UserMapper.findUserById “
column = ” user\_id “ >
<！ -实现对用户信息进行延迟加载- >
</ association >
</ resultMap >

非与延迟加载的主要区别就在association标签属性多了select状语从句：column

< association property = “ user ” javaType = “ com.iot.mybatis.po.User ”
select = “ com.iot.mybatis.mapper.UserMapper.findUserById ”
column = “ user\_id ” >

mapper.java

//查询订单关联查询用户，用户信息是延迟加载
public List < Orders > findOrdersUserLazyLoading（）抛出异常 ;

测试思路

执行上边映射器方法（findOrdersUserLazyLoading），内部去调用com.iot.mybatis.mapper.OrdersMapperCustom中的findOrdersUserLazyLoading只查询命令信息（单表）。

在程序中去遍历上一步骤查询出的列表中，当我们调用订单中的的getUser方法时，开始进行延迟加载。

延迟加载，去调用UserMapper.xml中findUserbyId这个方法获取用户信息。

延迟加载配置

MyBatis的默认没有开启延迟加载，需要在SqlMapConfig.xml中设置配置。

在MyBatis的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 '假'，则所有相关联的都会被初始化加载 | 真假 | 假 |
| aggressiveLazyLoading | 当设置为 '真' 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。 | 真假 | 真正 |

在SqlMapConfig.xml中配置：

< settings >
<！ -打开延迟加载的开关- >
< setting name = “ lazyLoadingEnabled ” value = “ true ” />
<！ -将积极加载改为消极加载即按需要加载- >
< 设置 名称 = “ aggressiveLazyLoading ” value = “ false ” />
<！ -开启二级缓存- >
<！ - <setting name =“cacheEnabled”value =“true”/> - >
</ settings >

测试代码

//查询订单关联查询用户，用户信息使用延迟加载
@Test
公共 空隙 testFindOrdersUserLazyLoading（）抛出异常 {
的SqlSession SQLSESSION = SqlSessionFactory中。的openSession（）; //创建代理对象
OrdersMapperCustom ordersMapperCustom = sqlSession
.getMapper（OrdersMapperCustom 。类）;
//查询订单信息（单表）
List < Orders > list = ordersMapperCustom 。findOrdersUserLazyLoading（）;
//遍历上边的订单列表
为（订单的订单：列表）{
//执行的getUser（）去查询用户信息，这里实现按需加载
用户用户 =订单。的getUser（）;
系统。出。的println（用户）;
}
}

延迟加载思考

不使用的MyBatis提供的关联及集合中的延迟加载功能，如何实现延迟加载？

实现方法如下：

定义两个映射方法：

查询订单列表

根据用户ID查询用户信息

实现思路：

先去查询第一个映射器方法，获取订单信息列表;在程序中（服务），按需去调用第二个映射器方法去查询用户信息。

总之，使用延迟加载方法，先去查询简单的SQL（最好单表，也可以关联查询），再去按需要加载关联查询的其它信息。

IDEA的调试小现象

使用的是intellij IDEA 15.0.2

先说一下结果吧，IDEA在调试和运行条件下，打印结果不同

我为了验证延迟加载前的用户是否为空，在订单类中加入了

public void print（）{
System 。出。println（“ ---- test-print ----- ” + user + “ user == null：” +（user == null））;
}

测试代码如下：

//查询订单关联查询用户，用户信息使用延迟加载
@Test
公共 空隙 testFindOrdersUserLazyLoading（）抛出异常 {
的SqlSession SQLSESSION = SqlSessionFactory中。的openSession（）; //创建代理对象
OrdersMapperCustom ordersMapperCustom = sqlSession
.getMapper（OrdersMapperCustom 。类）;
//查询订单信息（单表）
List < Orders > list = ordersMapperCustom 。findOrdersUserLazyLoading（）;
//遍历上边的订单列表
为（订单的订单：列表）{
//执行的getUser（）去查询用户信息，这里实现按需加载
//用户的用户= orders.getUser（）;
// System.out.println（user）;
订单。打印（）;
}
}

然后分别运行和调试

运行输出

DEBUG [main] - Opening JDBC Connection
DEBUG [main] - Created connection 110771485.
DEBUG [main] - Setting autocommit to false on JDBC Connection [com.mysql.jdbc.JDBC4Connection@69a3d1d]
DEBUG [main] - ==> Preparing: SELECT \* FROM orders
DEBUG [main] - ==> Parameters:
DEBUG [main] - <== Total: 3
----test-print-----null user==null: true
----test-print-----null user==null: true
----test-print-----null user==null: true

调试输出

在List<Orders> list = ordersMapperCustom.findOrdersUserLazyLoading();打断点，运行完这句，日志输出为：

DEBUG [main] - Opening JDBC Connection
DEBUG [main] - Created connection 1219273867.
DEBUG [main] - Setting autocommit to false on JDBC Connection [com.mysql.jdbc.JDBC4Connection@48aca48b]
DEBUG [main] - ==> Preparing: SELECT \* FROM orders
DEBUG [main] - ==> Parameters:
DEBUG [main] - <== Total: 3

但是你当开点list属性时，控制台又输出了，而且可以看到列表里的用户是有内容的

DEBUG [main] - ==> Preparing: SELECT \* FROM user WHERE id=?
DEBUG [main] - ==> Parameters: 1(Integer)
DEBUG [main] - <== Total: 1
DEBUG [main] - ==> Preparing: SELECT \* FROM user WHERE id=?
DEBUG [main] - ==> Parameters: 10(Integer)
DEBUG [main] - <== Total: 1

运行完所有程序，控制台输出为：

DEBUG [main] - Opening JDBC Connection
DEBUG [main] - Created connection 1219273867.
DEBUG [main] - Setting autocommit to false on JDBC Connection [com.mysql.jdbc.JDBC4Connection@48aca48b]
DEBUG [main] - ==> Preparing: SELECT \* FROM orders
DEBUG [main] - ==> Parameters:
DEBUG [main] - <== Total: 3
DEBUG [main] - ==> Preparing: SELECT \* FROM user WHERE id=?
DEBUG [main] - ==> Parameters: 1(Integer)
DEBUG [main] - <== Total: 1
DEBUG [main] - ==> Preparing: SELECT \* FROM user WHERE id=?
DEBUG [main] - ==> Parameters: 10(Integer)
DEBUG [main] - <== Total: 1
----test-print-----User [id=1, username=王五, sex=2, birthday=null, address=null] user==null: false
----test-print-----User [id=1, username=王五, sex=2, birthday=null, address=null] user==null: false
----test-print-----User [id=10, username=张三, sex=1, birthday=Thu Jul 10 00:00:00 CST 2014, address=北京市] user==null: false

所以，我觉得应该是在调试时，查看属性的话，IDEA会自动调用得到相应的方法，从而触发用户的查询。延迟加载的源码实现以后我会阅读，把这个问题弄清楚。