# 本篇是本人看完所有Mybatis教程视频后总结的内容

# Mybatis框架学习4——缓存与延迟加载

## 主要学习内容：

Mybatis的延迟加载

Mybatis的两级缓存机制

Mybatis注解实现复杂的配置

## Mybatis的延迟加载

在上一部分中，学习了Mybatis的一对一、一对多及多对多关系的关联查询，但是在实际开发过程中很多时候我们并不需要总是在加载用户信息时就一定要加载他的账户信息，因此在开发过程中就可以使用Mybatis提供的延迟加载。

延迟加载就是在需要用到数据时才进行加载，不需要用到数据时就不加载数据。延迟加载也称懒加载。因此采用延迟加载时，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。

* 1. 延迟加载的设置

当我们查询账户和相关的用户时，使用<association>标签来完成关联查询，配置的resultMap如下。

<resultMap id="accountUserMap" type="account">

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

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

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

<association property="user" column="uid" javaType="user" select="com.itheima.dao.IUserDao.findById"></association>

</resultMap>

其中，select： 填写我们要调用的 select 映射的 id

column ： 填写我们要传递给 select 映射的参数

另外需要的操作就是在SqlMapConfig.xml文件中添加开启延迟加载的配置。

<settings>

<!--开启Mybatis支持延迟加载-->

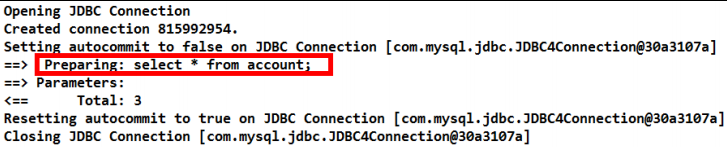
<setting name="lazyLoadingEnabled" value="true"/>

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

</settings>

在配置文件中配置了如上信息后，Mybatis的延迟加载就开启了。其中，aggressiveLazyLoading为false表示在进行关联查询后，每使用一个属性就加载哪个属性。

测试：当我们使用关联语句查询账户时，可以看到测试结果：



虽然账户类中包含了用户类，但是可以看到只进行了account的查询而没有进行User的查询。

同样，当我们想要查询用户对象和该用户拥有的所有账户信息时，可以使用<collection>标签来实现一对多的关联查询并完成延迟加载。配置的resultMap如下：

<mapper namespace="com.itheima.dao.IUserDao">

<resultMap id="userAccountMap" 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>

<collection property="accounts" ofType="account" select="com.itheima.dao.IAccountDao.findAccountByUid" column="id"></collection>

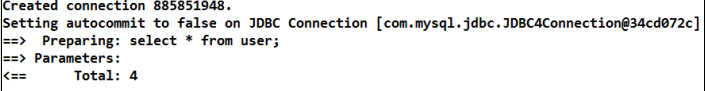
</resultMap>

在<collection>标签中，ofType 用于指定集合元素的数据类型

select 是用于指定查询账户的唯一标识（账户的 dao 全限定类名加上方法名称）

column 是用于指定使用哪个字段的值作为条件查询

同样，当我们查询所有用户时，我们可以发现并没有加载Account信息。



* 1. 小结

在这部分中，学习了如何用延迟加载策略来提高持久层操作的效率，在SqlMapConfig.xml中配置打开延迟缓存之后，通过持久层接口映射文件resultMap中的<association>和<collection>标签来实现支持一对一、一对多和多对多的延迟加载。

## 2、Mybatis 缓存

**2.1 Mybatis一级缓存**

一级缓存是 SqlSession 级别的缓存，只要 SqlSession 没有 flush 或 close，它就存在。

mybatis的一级缓存不需要手动打开，而是自身存在一级缓存。

**2.1.1一级缓存的测试**

**首先，使用查找用户方法来测试Mybatis的一级缓存。**

<select id="findById" parameterType="INT" resultType="user" useCache="true">

select \* from user where id = #{uid}

</select>

**在测试文件中编写：**

**User** user1 = userDao.findById(41);

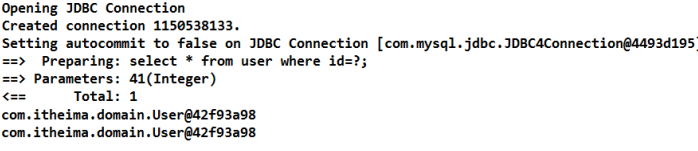
**System**.out.println(user1);

**User** user2 = userDao.findById(41);

**System**.out.println(user2);

**System**.out.println(user1 == user2);

**在测试中我们对id为41的用户进行了两次查询，结果如下：**



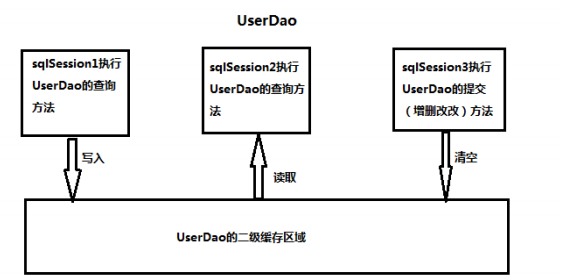
发现，虽然在上面的代码中查询了两次，但最后只执行了一次数据库操作，这就是 Mybatis 提供的一级缓存在起作用。因为一级缓存的存在，导致第二次查询 id 为 41 的记录时，并没有发出sql语句从数据库中查询数据，而是从同一个session的一级缓存中查询，因此两次查询出来的User是同一个User。

**2.1.2一级缓存的清除**

如果sqlSession执行了查询操作之后，执行commit操作（执行插入、更新、删除），则**Mybatis会清空 SqlSession 中的一级缓存**，这样做的目的为了让缓存中存储的是最新的信息，避免脏读。

**2.2 Mybatis二级缓存**

**与一级缓存不同的是，Mybatis的二级缓存是存在于sessionfactory中的。**二级缓存是 mapper 映射级别的缓存，多个 SqlSession 去操作同一个 Mapper 映射的 sql 语句，多个 SqlSession 可以共用二级缓存，二级缓存是跨 SqlSession 的。



**2.2．1 二级缓存的设置**

**与一级缓存不同的是，二级缓存需要手动开启，即Mybatis默认是关闭二级缓存的。**

**首先在SqlMapConfig.xml中开启二级缓存：**

<settings>

<setting name="cacheEnabled" value="true"/>

</settings>

**第二步，在相关持久层接口的Mapper映射文件中加上二级缓存，即加上**

<cache/>标签

**第三步，**在statement上添加上useCache=”true”这个属性值，表示这个statement要使用二级缓存**。注意：针对每次查询都需要最新的数据 sql，要设置成 useCache=false，禁用二级缓存。**

**2.2.2二级缓存的测试**

**在测试中使用如下代码**

**SqlSession** sqlSession1 = factory.openSession();

**IUserDao** dao1 = sqlSession1.getMapper(**IUserDao**.class);

**User** user1 = dao1.findById(41);

**System**.out.println(user1);

sqlSession1.close();//一级缓存消失

**SqlSession** sqlSession2 = factory.openSession();

**IUserDao** dao2 = sqlSession2.getMapper(**IUserDao**.class);

**User** user2 = dao2.findById(41);

**System**.out.println(user2);

sqlSession2.close();

**System**.out.println(user1 == user2);

当我们分别用同一个factory的两个sqlSession来进行查询的时候，我们发现用第二个sqlSession语句查询并没有在数据库中使用sql查询语句，说明第二次的查询使用了二级缓存。**但是需要注意的是，两次查询出来的user1和user2并不是同一个，因为存在factory中的二级缓存是数据而不是类，因此两个并不是同一个类。因此我们要注意，当我们在使用二级缓存时，所缓存的类一定要实现 java.io.Serializable 接口，这种就可以使用序列化 方式来保存对象**。

2.3小结

Mybatis给我们提供了两级的缓存。mybatis一级缓存存在sqlsession中，存的是对象，不用开启。二级缓存存在sessionFactory中，存的是数据，需要手动开启。当对数据库执行了增删改操作时，缓存的内容会消失。

## 3、注解方式代替映射配置文件

单表的 CRUD 操作是最基本的操作，前面的学习都是基于 Mybaits 的映射文件来实现的。

**3.1 使用注解方式开发持久层接口**

**3.1.1 CRUD注解**

**首先，我们能使用@Select、@Insert等等注解来声明当前方法执行的是哪种操作。**

@Select("select \* from user")

**List**<**User**> **findAll**();

@Insert("insert into user(username,address,sex,birthday)values(#{username},#{address},#{sex},#{birthday})")

**void** **saveUser**(**User** user);

**3.1.2 resultMap注解**

**可以使用Results注解来定义resultMap，并使用@ResultMap注解来使用resultMap,完成数据库列名和实体属性名的一一对应。**

@Results(id="userMap",value={

@Result(id=**true**,column = "id",property = "userId"),

@Result(column = "username",property = "userName"),

@Result(column = "address",property = "userAddress"),

@Result(column = "sex",property = "userSex"),

@Result(column = "birthday",property = "userBirthday"),

@Result(property = "accounts",column = "id"})

**因此通过注解Dao接口，就不需要dao接口的配置文件来完成映射了。**

**3.1.3 SqlMapConfig.xml文件配置**

**需要注意的是，**当在Dao接口中使用了注解映射时，不需要映射配置xml文件也能完成映射，在SqlMapConfig.xml文件的<mappers>标签中使用package或class即可。但不管mapper中用package还是class，如果此时存在相应的xml文件则会报错，即使用了注解后只要有接口.xml映射文件就会报错。所以一个dao要么用注解，要么用xml。通常来说所有dao都统一用一种方式。

**3.2注解实现复杂关联查询**

**同样，首先，注解方式使用注解**@Results来代替xml文件中的<resultMap>标签。

**第二，使用注解@Result代替<resultMap>中的<id>和<result>，@Result中的**id表示是否是主键字段，column表示数据库的列名，property表示需要装配的属性名。

其次，使用@One注解代替<association>标签完成一对一的关联查询，在注解中用来指定子查询返回单一对象。使用@Many注解代替<colleciton>标签完成一对多的关联查询，在注解中用来指定子查询返回对象集合。参考代码如下：

@Results(id="userMap",value={

@Result(id=**true**,column = "id",property = "userId"),

@Result(column = "username",property = "userName"),

@Result(column = "address",property = "userAddress"),

@Result(column = "sex",property = "userSex"),

@Result(column = "birthday",property = "userBirthday"),

@Result(property = "accounts",column = "id",

many = @Many(select = "com.itheima.dao.IAccountDao.findAccountByUid",

fetchType = **FetchType**.LAZY))

})

其中，fetchType相当于开启lazyLoadingEnabled延迟加载。

**3.3基于注解的二级缓存**

同样，基于注解开启二级缓存首先需要在SqlMapConfig.xml中开启二级缓存。

<settings>

<setting name="cacheEnabled" value="true"/>

</settings>

接着即在Dao接口中使用注解开启二级缓存。

在需要开启的Dao接口前添加注解@CacheNamespace(blocking=true)

即可使该接口中的方法使用Mybatis二级缓存。

3.4 小结

本节是使用注解的方法来完全实现Dao接口映射配置文件所实现的功能，如复杂的多表关联查询，一对一、一对多等等，延迟加载，以及二级缓存，基本所有功能都可改用注解实现，但是需要注意注解和xml文件只能存在一个，否则会出现报错，而且在实际开发情况下，所有的Dao都使用同一种映射方法。

## 5、总结

总的来说，这部分学习内容介绍了Mybatis提供的延迟加载和两级缓存机制，并学习了这些机制的配置过程与使用方法，同时还用注解方法实现了本部分所讲的延迟加载和缓存、以及上一部分的复杂的多表关联查询，可以完全的替代了xml映射文件。