# mybatis 使用 PageHelper 实现一对多正确分页

在联表查询的时候，我比较喜欢使用 LEFT JOIN，这在一对一的关系中，在 Mybatis Mapper XML 中使用 <association> 标签是没有问题的。

但是昨晚在拆分表后发现分页竟然不对了。业务是模拟朋友圈发表状态，原来发表状态的可以加一张图片，所以图片的地址直接放在状态表中了。然后我希望在发表状态的时候可以最多加 9 张图片，形成九宫格的样子。

所以就把表拆分了，状态表为 tweet， 状态图片表为 tweet\_image。然后使用左连接查询加 <collection> 标签，就发现不对了。

为了简要说明，我简化了一下表。

CREATE TABLE `tweet` (

`id` bigint(20) NOT NULL,

`content` varchar(1024) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8

CREATE TABLE `tweet\_image` (

`id` bigint(20) NOT NULL,

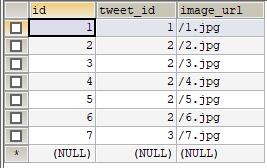
`tweet\_id` bigint(20) DEFAULT NULL, -- 状态表外键

`image\_url` varchar(1024) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8





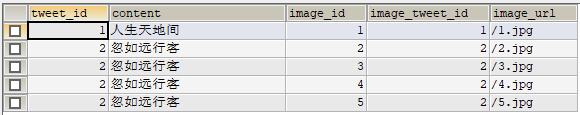
Demo 里有 3 条状态，每条状态都有不等的图片附件数量。加入分页大小是 5， 那么下面的 SQL 在 PageHelper 中返回的分页大小是多少呢？

SELECT

tweet.id AS tweet\_id, tweet.content,

image.id AS image\_id, image.tweet\_id AS image\_tweet\_id, image.image\_urlFROM tweet AS tweetLEFT JOIN tweet\_image AS image ON image.tweet\_id = tweet.idLIMIT 5

结果只返回了 2 条状态，并且 id 为 2 的状态图片附件数量也少了，这就是问题所在。



问题分析到这里就很清楚了，下面就说一下具体的解决方案。

## PageHelper 配置

Maven 依赖：

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

</dependency>

<dependency>

<groupId>com.github.pagehelper</groupId>

<artifactId>pagehelper</artifactId>

<version>5.1.4</version>

</dependency></dependencies>

在 spring-context.xml 中配置 PageHelper 插件：

<bean id="sessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

<property name="dataSource" ref="dataSource"/>

<property name="typeAliasesPackage" value="com.github.\*\*.domain"/>

<property name="configLocation" value="classpath:mybatis-config.xml"/>

<property name="mapperLocations" value="classpath:mappings/\*\*/\*.xml"/>

<property name="plugins">

<array>

<bean class="com.github.pagehelper.PageInterceptor">

<!-- 这里的几个配置主要演示如何使用，如果不理解，一定要去掉下面的配置 -->

<property name="properties">

<value>

helperDialect=mysql

reasonable=true

autoRuntimeDialect=true

</value>

</property>

</bean>

</array>

</property></bean>

## 实体类

tweet 表和 tweet\_image 表的实体类。

public class **Tweet** implements **Serializable**{

private Long id;

private String content;

private List<TweetImage> images;

// 省略 getter setter toString 方法

}

public class **TweetImage** implements **Serializable**{

private Long id;

private String url;

private Tweet tweet;

// 省略 getter setter toString 方法

}

## Service

其实 Service 方法和平常一样，不需要做改变。

public PageInfo<Tweet> **getPage**(Integer pageNumber){

PageHelper.startPage(pageNumber, WebConstants.PAGE\_SIZE);

List<Tweet> tweets = tweetMapper.getPage();

return new PageInfo<>(tweets);

}

## Mapper XML

Mapper 接口就没什么好说的了。

/\*\*

\* 获取 Tweet 分页列表

\*/List<Tweet> **getPage**();

Mapper 映射文件 TweetMapper.xml，所有一对多关系都是懒加载的：

<resultMap id="TweetEntity" type="Tweet">

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

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

<collection property="images" // Tweet 实体中的 images List

column="id" // tweet 表 id

ofType="TweetImage" // collection 元素类型

javaType="java.util.List" // 查询返回的类型为图片的 List 列表

select="getImageByTweetId"/> // 指向的 select 查询</resultMap>

<!-- collection 查询 --><select id="getImageByTweetId" parameterType="long" resultType="map">

SELECT

image.id, image.url

FROM

tweet\_image AS image

WHERE

// 这里传值进来的 id 是 getPage 中 id 的列名，

// 即 TweetEntity 中的 column 属性而不是 property 属性名

image.tweet\_id = #{id}</select>

<!-- 获取 Tweet 分页列表 --><select id="getPage" resultMap="TweetEntity">

SELECT

tweet.id, tweet.content

FROM

tweet AS tweet</select>

## 查询结果

这次再次刷新，分页大小为 5，返回数据就正确了。

[

{

"id": 1,

"content": "人生天地间",

"images": [{

"id": "1",

"url": "/1.jpg"

}

]

},

{

"id": 2,

"content": "忽如远行客",

"images": [{

"id": "2",

"url": "/2.jpg"

},

{

"id": "3",

"url": "/3.jpg"

},

{

"id": "4",

"url": "/4.jpg"

},

{

"id": "5",

"url": "/5.jpg"

},

{

"id": "6",

"url": "/6.jpg"

}

]

},

{

"id": 3,

"content": "秋天来了",

"images": [{

"id": "7",

"url": "/7.jpg"

}

]

}

]

好了，完美！

其实 MyBatis 一对多的分页写法还有很多的，或者上面的配置稍微调整一下都可以。