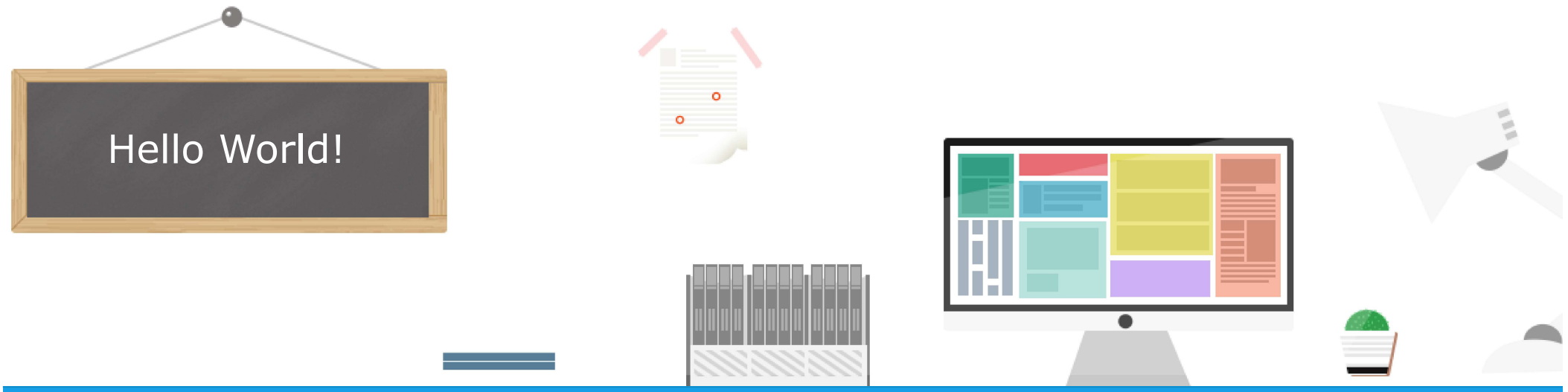

mybatis查询树形数据的两种方法

最近开发中遇到了很多树形结构数据的需要，利用mybatis提供嵌套查询功能，基本上可以完美解决，但是对于其中的原理并不理解，导致在使用的时候像瞎猫碰死耗子一样，照着先前成功的例子copy，后来遇到了莫名奇怪的报错迟迟不能解决，于是百度了一番，大致了解了背后的原理，整理如下。

以简单的角色-菜单为例

表结构

menu		
id	int(14)	<pk>
name	varchar(255)	
description	text	
parent	int(14)	
createdate	datetime	
modifydate	datetime	

role		
id	int(14)	<pk>
name	varchar(255)	
description	text	
createdate	datetime	
modifydate	datetime	

roleandmenu		
id	int(14)	<pk>
roleId	int(14)	
menuId	int(14)	

其中menu为菜单表，role为角色表，roleandmenu是中间表，角色和菜单为多对多的关系，现在我们需要下图所示的实体类

```
1 import java.util.List;
2
3 public class RoleInfo {
4
5     private Integer roleid;
6     private String rolename;
7     private List<Menu> menulist;
8     public Integer getRoleid() {
9         return roleid;
10    }
11    public void setRoleid(Integer roleid) {
12        this.roleid = roleid;
13    }
14    public String getRolename() {
15        return rolename;
16    }
17    public void setRolename(String rolename) {
18        this.rolename = rolename;
19    }
20    public List<Menu> getMenulist() {
21        return menulist;
22    }
23    public void setMenulist(List<Menu> menulist) {
24        this.menulist = menulist;
25    }
26
27    @Override
28    public String toString() {
29        return "RoleInfo [roleid=" + roleid + ", rolename=" + rolename + ", menulist=" + menulist + "]";
30    }
```

```
31
32 }
```

第一种方法:利用嵌套语句查询

```
<resultMap type="com.test.mybatis.model.RoleInfo" id="roleModel">
        <id column="id" property="roleid"/>
        <result column="name" property="rolename"/>
        <collection property="menulist" select="getMenu" column="id">

        </collection>
    </resultMap>

    <select id="getRoleInfo" resultMap="roleModel">
        select id,name from role
    </select>

    <select id="getMenu" resultType="com.test.mybatis.model.Menu">
        select m.id,m.name
        from menu m join roleandmenu ram on m.id=ram.menuId
        where ram.roleId=#{id}
    </select>
```

```
    @Test
    public void testRoleAndMenu() throws IOException {
        Reader reader = Resources.getResourceAsReader("mybatis.xml");
        SqlSessionFactory sqlsessionfac = new SqlSessionFactoryBuilder().build(reader);
        SqlSession sqlsession = sqlsessionfac.openSession();
        try {
            RoleAndMenuMapper mapper = sqlsession.getMapper(RoleAndMenuMapper.class);
            System.out.println(JSONObject.toJSON(mapper.getRoleInfo()));
        } catch (Exception e) {
            // TODO: handle exception
            e.printStackTrace();
        } finally {
            sqlsession.close();
        }
    }
```

结果:


```
[
  {
    "menulist":[
      {
        "name":"菜单1",
        "id":1
      },
      {
        "name":"菜单2",
        "id":2
      }
    ],
    "roleid":1,
    "rolename":"角色1"
  },
  {
    "menulist":[
      {
        "name":"菜单3",
        "id":3
      },
      {
        "name":"菜单4",
        "id":4
      }
    ],
    "roleid":2,
    "rolename":"角色2"
  },
```


原理如下:

1.mybatis先执行**getRoleInfo**这个查询，获取结果集

2.从ResultSet中逐一取出记录，构建**RoleInfo**对象并为映射属性赋值

3.赋值过程中发现目标**menulist**属性配置了一个关联集合（collection），此时执行id为collection标签中select属性值（**getMenu**）的查询，**并将当前记录中的id属性作为此查询的参数。（association标签同理）**

1. mybatis查询树形数据的两种方法(5288)

2. mybatis 空字符串和0(506)

3. Java导出Excel，java.io.IOException: Stream is already closed(33)

4.将关联查询返回的结果映射到**meunlist**属性

5.执行步骤2，直至**ResultSet.next=false**

6.返回查询结果

这种方式的好处在于简单易懂，通过简单的配置就可以达到目标效果。不足之处在于如果结果集记录条数过大，会造成较大的数据库访问消耗，因为在从ResultSet中取出记录的时候每取一条，便执行一次关联查询，假设一次查询的结果集有10条记录，则数据库的访问次数为：**关联查询次数**(10)+**返回结果集的查询**(1)=11次。

需要注意的地方

1.collection/association标签的**column**属性：当向关联查询传递的参数个数为1时，column的值应为结果集中的列名，而不是映射属性名（property），上面的例子中，向关联查询传递**id**值，column的值应为**id**而不是**roleid。**

可以向关联查询传递多个参数，此时column的值为多个键值对，如下图

```
<resultMap type="com.test.mybatis.model.RoleInfo" id="roleModel">
    <id column="id" property="roleid"/>
    <result column="name" property="rolename"/>
    <collection property="menulist" select="getMenu" column="{id=id,name=name}">

    </collection>
</resultMap>
```

此时向关联查询传递了两个参数id和name，此时还应该将关联的查询的parameterType改为java.util.Map，否则关联查询无法接受参数

```
<select id="getMenu" parameterType="java.util.Map"
     resultType="com.test.mybatis.model.Menu">
    select m.id,m.name
    from menu m join roleandmenu ram on m.id=ram.menuId
    where ram.roleId=#{id}
</select>
```

2.在进行单一类型树形结构查询的时候，需要注意关联查询的结果集中的列是否有作为查询条件的列

这样说可能比较别扭，以上面的**menu**表为例，有一个**parent**列用于存储父部门的ID，使用嵌套查询获取以下实体类

```
import java.util.List;

public class MenuTree {

    private Integer id;
    private String name;
    private List<Menu> children;
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public List<Menu> getChildren() {
        return children;
    }
    public void setChildren(List<Menu> children) {
        this.children = children;
    }


}
```

```
<resultMap type="com.test.mybatis.model.MenuTree" id="menuTreeModel">
    <id column="id" property="id"/>
    <result column="name" property="name"/>
    <collection property="children" select="getSubMenu" column="id">

    </collection>
</resultMap>

<select id="getMenuTree" resultMap="menuTreeModel">
    select id,name from menu
    where parent is null
</select>

<select id="getSubMenu" resultMap="menuTreeModel">
    select name from menu
    where parent=#{id}
</select>
```

此时会产生一个问题：每个顶级菜单(parent为空的菜单)的子菜单只有一个查询结果，这是因为在关联查询getSubMenu中没有将id查询出来，而关联查询和主查询的resultMap一样，所以关联查询在映射结果集的时候就会再次去执行关联查询，而由于本次关联查询并没有取出id这个作为参数的属性，所以实际上只执行了N（结果集记录数）次关联查询。

因此，在这种情况中，必须在关联查询中查询出id这个列，否则会查询不出预期结果。

```
<select id="getSubMenu" resultMap="menuTreeModel">
    select name from menu
    where parent=#{id}
</select>
```

错误

```
<select id="getSubMenu" resultMap="menuTreeModel">
    select id,name from menu
    where parent=#{id}
</select>
```

正确

第二种方法：使用嵌套结果集

```
<resultMap type="com.test.mybatis.model.RoleInfo" id="roleModel">
        <id column="id" property="roleid"/>
        <result column="name" property="rolename"/>
        <collection property="menulist" ofType="com.test.mybatis.model.Menu">
            <id column="menuid" property="id"/>
            <result column="menuname" property="name"/>
            <result column="description" property="description"/>
            <result column="parent" property="parent"/>
            <result column="createdate" property="createdate"/>
            <result column="modifydate" property="modifydate"/>
        </collection>
    </resultMap>

    <select id="getRoleInfo" resultMap="roleModel">
        select
            ram.roleid as id,
            ro.name as name,
            me.id as menuid,
            me.name as menuname,
            me.description,
            me.parent,
            me.createdate,
            me.modifydate
        from roleandmenu ram
            left outer join role ro on ram.roleid=ro.id
            left outer join menu me on ram.menuid=me.id
    </select>
```

原理是通过关联查询，一次性将数据查询出来，然后根据resultMap的配置进行转换，构建目标实体类。

显然，这种方法更为直接，只需要访问一次数据库就可以了，不会造成严重的数据库访问消耗。

但是以上是查询出全部数据的情况，因为只有查询出全部数据，才能得到最终结果。如果直接分页的话，会导致数据被截断，也就是collection中的数据残缺。

这种情况最好的处理方式就是手动分页，对主表分页，对其他连接的表不分页。

将上面的SQL改为

```
SELECT
    base.id,
    ro. NAME AS NAME,
    me.id AS menuid,
    me. NAME AS menuname,
    me.description,
    me.parent,
    me.createdate,
    me.modifydate
FROM
    (
        SELECT
            roleid AS id,
            menuid
        FROM
            roleandmenu
        LIMIT $ { pageNum }, $ { pageSize }
    ) AS base
LEFT OUTER JOIN role ro ON base.id = ro.id
LEFT OUTER JOIN menu me ON base.menuid = me.id
```

这里手动传入**pageNum**,**pageSize**，对主表roleandmenu分页，从表role和menu不分页。

这样就可以得到正确的数据。

此外，还有一种情况，如果把嵌套结果集的返回值类型全部改成**HashMap**的话，会导致**menulist**里只有一行数据

```
<resultMap type="java.util.HashMap" id="roleModel">
    <id column="id" property="roleid"/>
    <result column="name" property="rolename"/>
    <collection property="menulist" ofType="java.util.HashMap">
        <id column="menuid" property="id"/>
        <result column="menuname" property="name"/>
        <result column="description" property="description"/>
        <result column="parent" property="parent"/>
        <result column="createdate" property="createdate"/>
        <result column="modifydate" property="modifydate"/>
    </collection>
</resultMap>
```

解决的办法是，给**collection**标签的**javaType**赋值为目标集合类型

```
<resultMap type="java.util.HashMap" id="roleModel">
    <id column="id" property="roleid"/>
    <result column="name" property="rolename"/>
    <collection property="menulist" ofType="java.util.HashMap" javaType="java.util.List">
        <id column="menuid" property="id"/>
        <result column="menuname" property="name"/>
        <result column="description" property="description"/>
        <result column="parent" property="parent"/>
        <result column="createdate" property="createdate"/>
        <result column="modifydate" property="modifydate"/>
    </collection>
</resultMap>
```

找了很久，终于在官方文档里找到了解释

属性	描述
`column`	来自数据库的类名,或重命名的列标签。这和通常传递给 resultSet.getString(columnName)方法的字符串是相同的。
`javaType`	一个 Java 类的完全限定名,或一个类型别名(参考上面内建类型别名的列表)。如果你映射到一个 JavaBean,MyBatis 通常可以断定类型。然而,如 果你映射到的是 HashMap,那么你应该明确地指定 javaType 来保证所需的 行为。
`jdbcType`	在这个表格之前的所支持的 JDBC 类型列表中的类型。JDBC 类型是仅仅 需要对插入, 更新和删除操作可能为空的列进行处理。这是 JDBC 的需要, jdbcType 而不是 MyBatis 的。如果你直接使用 JDBC 编程,你需要指定这个类型-但 仅仅对可能为空的值。
`typeHandler`	我们在前面讨论过默认的类型处理器。使用这个属性,你可以覆盖默认的 类型处理器。 这个属性值是类的完全限定名或者是一个类型处理器的实现, 或者是类型别名。
`select`	用于加载复杂类型属性的映射语句的ID,从column中检索出来的数据，将作为此select语句的参数。具体请参考Association标签。
`resultMap`	ResultMap的ID，可以将嵌套的结果集映射到一个合适的对象树中，功能和select属性相似，它可以实现将多表连接操作的结果映射成一个单一的 `ResultSet` 。这样的 `ResultSet` 将会将包含重复或部分数据重复的结果集正确的映射到嵌套的对象树中。为了实现它, MyBatis允许你 “串联” ResultMap,以便解决嵌套结果集的问题。想了解更多内容，请参考下面的Association元素。
`name`	构造方法形参的名字。通过指定具体的名字你可以以任意顺序写入arg元素。参看上面的解释。从3.4.3版本起。

参考资料:http://www.jianshu.com/p/9e397d5c85fd

标签: mybatis