# 案例：商品的类别

**【案例目标】**

现有一个商品表product和一个商品类别表category，其中，商品类别表category和商品表product是一对多的关系。商品表product和商品类别表category详情分别如表4-1和表4-2所示。

1. 商品表（product）详情

|  |  |  |  |
| --- | --- | --- | --- |
| **商品编号**  **（id）** | **商品名称**  **（goodsname）** | **商品单价**  **（price）** | **商品类别**  **（typeid）** |
| 1 | 电视机 | 5000 | 1 |
| 2 | 冰箱 | 4000 | 2 |
| 3 | 空调 | 3000 | 2 |
| 4 | 洗衣机 | 2000 | 2 |

1. 商品类别表（category）详情

|  |  |
| --- | --- |
| **商品类别编号**  **（id）** | **商品类别名称**  **（typename）** |
| 1 | 黑色家电 |
| 2 | 白色家电 |

本案例具体要求如下。

根据表1和表2在数据库分别创建一个商品表product和一个商品类别表category， 并通过MyBatis查询商品类别为白色家电的商品的所有信息。

**【实现步骤】**

**1.项目搭建**

创建一个名称为mybatis-demo04的项目，项目的具体搭建过程请参考1.3节。

**2.数据库准备**

在名为mybatis的数据库中，创建两个数据表，分别为product和category，同时在表中预先插入几条测试数据。具体的SQL语句如下所示。

USE mybatis;

# 创建一个名称为category的表

CREATE TABLE category (

id int(32) PRIMARY KEY AUTO\_INCREMENT,

typename varchar(40)

);

# 插入2条数据

INSERT INTO category VALUES (1, '黑色家电');

INSERT INTO category VALUES (2, '白色家电');

# 创建一个名称为product的表

CREATE TABLE product (

id int(32) PRIMARY KEY AUTO\_INCREMENT,

goodsname varchar(40),

price DOUBLE,

category\_id int(32) NOT NULL,

FOREIGN KEY(category\_id) REFERENCES category(id)

);

# 插入4条数据

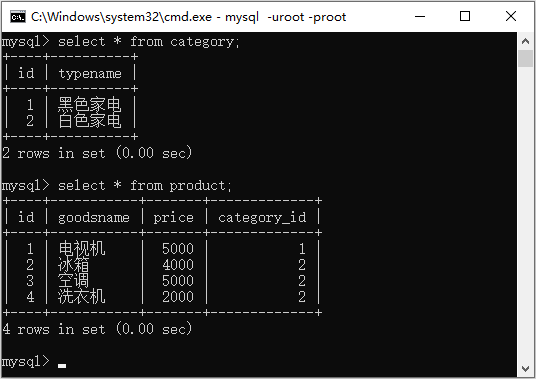
INSERT INTO product VALUES (1, '电视机', 5000,1);

INSERT INTO product VALUES (2, '冰箱', 4000,2);

INSERT INTO product VALUES (3, '空调', 5000,2);

INSERT INTO product VALUES (4, '洗衣机', 2000,2);

执行完上述操作后，category表和product表中的数据如图4-1所示。



1. category和product表中的数据

**3.POJO类准备**

（1）在com.itheima.pojo包中，创建持久化类Category，并在类中定义商品类别的相关属性和方法。Category类具体代码如文件4-1所示。

1. Category.java
2. package com.itheima.pojo;
3. import java.util.List;
4. /\*\*
5. \* 商品类别持久化类
6. \*/
7. public class Category {
8. private Integer id; // 主键id
9. private String typename; // 类别名称
10. private List<Product> productList; // 商品集合
11. public Integer getId() {
12. return id;
13. }
14. public void setId(Integer id) {
15. this.id = id;
16. }
17. public String getTypename() {
18. return typename;
19. }
20. public void setTypename(String typename) {
21. this.typename = typename;
22. }
23. public List<Product> getProductList() {
24. return productList;
25. }
26. public void setProductList(List<Product> productList) {
27. this.productList = productList;
28. }
29. @Override
30. public String toString() {
31. return "Category{" +
32. "id=" + id + ", typename='" + typename +
33. ", productList=" + productList + '}';
34. }
35. }

（2）在com.itheima.pojo包中，创建持久化类Product，并在类中定义相关属性和方法。Product类具体代码如文件4-2所示。

1. Product.java
2. package com.itheima.pojo;
3. /\*\*
4. \* 商品持久化类
5. \*/
6. public class Product {
7. private Integer id; // 主键id
8. private String goodsname; // 商品名称
9. private double price; // 价格
10. public Integer getId() {
11. return id;
12. }
13. public void setId(Integer id) {
14. this.id = id;
15. }
16. public String getGoodsname() {
17. return goodsname;
18. }
19. public void setGoodsname(String goodsname) {
20. this.goodsname = goodsname;
21. }
22. public double getPrice() {
23. return price;
24. }
25. public void setPrice(double price) {
26. this.price = price;
27. }
28. @Override
29. public String toString() {
30. return "Product{" +
31. "id=" + id + ", goodsname='" + goodsname +
32. ", price=" + price + '}';
33. }
34. }

**4.编写映射文件**

在com.itheima.mapper包中，创建商品类别实体映射文件CategoryMapper.xml，并在文件中编写一对多关联映射查询的配置。CategoryMapper.xml具体配置如文件4-3所示。

1. CategoryMapper.xml
2. <?xml version="1.0" encoding="UTF-8"?>
3. <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4. "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5. <!-- namespace表示命名空间 -->
6. <mapper namespace="com.itheima.mapper.CategoryMapper">
7. <!-- 一对多：查看某一商品类别及其关联的商品信息
8. 注意：当关联查询出的列名相同，则需要使用别名区分 -->
9. <select id="findCategoryWithProduct" parameterType="Integer"
10. resultMap="CategoryWithProductResult">
11. SELECT c.\*,p.id as category\_id,p.goodsname,p.price
12. from category c,product p
13. WHERE c.id=p.category\_id
14. and c.id=#{id}
15. </select>
16. <resultMap type="com.itheima.pojo.Category"
17. id="CategoryWithProductResult">
18. <id property="id" column="id"/>
19. <result property="typename" column="typename"/>
20. <!-- 一对多关联映射：collection
21. ofType表示属性集合中元素的类型，List<Product>属性即Product类 -->
22. <collection property="productList"
23. ofType="com.itheima.pojo.Product">
24. <id property="id" column="category\_id"/>
25. <result property="goodsname" column="goodsname"/>
26. <result property="price" column="price"/>
27. </collection>
28. </resultMap>
29. </mapper>

在文件4-3中，使用了MyBatis嵌套结果的方式定义了一个根据商品类别id查询商品类别及其关联的商品信息的select语句。因为返回的商品类别对象中，包含Product集合对象属性，所以需要手动编写结果映射信息。其中，第8~14行代码写了一条复杂的多表关联的SQL语句（关联查询商品类别及其对应的商品信息）；第15~27行代码在<collection>元素中使用相关子元素进行数据库表字段和实体类属性的一一映射。

**5.修改mybatis-config.xml核心配置文件**

在核心配置文件mybatis-config.xml中，引入CategoryMapper.xml，将CategoryMapper.xml映射文件加载到程序中。在mybatis-config.xml中的<mappers>元素下添加如下代码。

<mapper resource="com/itheima/mapper/CategoryMapper.xml" />

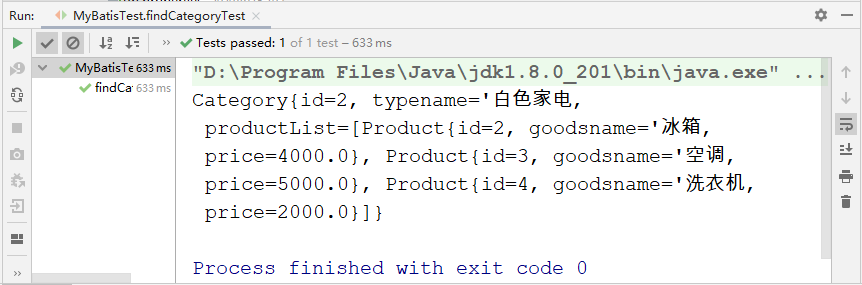
**6.编写测试方法**

在测试类MyBatisTest中，编写测试方法findCategoryTest()，其代码如下所示。

1. /\*\*
2. \* 一对多
3. \*/
4. @Test
5. public void findCategoryTest() {
6. // 1.通过工具类生成SqlSession对象
7. SqlSession session = MyBatisUtils.getSession();
8. // 2.查询id为2的商品类别信息
9. Category category = session.selectOne("com.itheima.mapper."
10. + "CategoryMapper.findCategoryWithProduct", 2);
11. // 3.输出查询结果信息
12. System.out.println(category);
13. // 4.关闭SqlSession
14. session.close();
15. }

在上述代码中，第7行代码通过MyBatisUtils工具类获取SqlSession对象；第9~10行代码通过SqlSession对象调用selectOne()方法获取id为2的商品类别信息。第12行代码使用输出语句输出查询结果信息；第14行代码关闭SqlSession，释放资源。

执行MyBatisTest测试类的findCategoryTest()方法，控制台的输出结果如图4-2所示。



1. findCategoryTest()方法运行结果

由图4-2输出结果分析可知，使用MyBatis查询出了商品类别及其关联的商品集合信息。实现了MyBatis一对多的关联查询。