

Hibernate多表关系

[Hibernate多表关系](#)

[课前默写](#)

[课程回顾](#)

[今天任务](#)

[教学目标](#)

[第十章 一对多映射](#)

[10.1 一对多实现](#)

[10.1.1 准备项目](#)

[10.1.2 创建订单表](#)

[10.1.3 创建Order实体类](#)

[10.1.4 修改Customer实体类](#)

[10.1.5 Customer配置一对多](#)

[10.1.6 Order配置多对一](#)

[10.1.7 将映射文件加入hibernate.cfg.xml](#)

[10.1.8 测试新增关联数据](#)

[10.1.9 测试查询订单](#)

[10.2 cascade级联操作](#)

[10.2.1 级联保存](#)

[10.2.2 级联删除](#)

[10.3 inverse关系反转](#)

[10.3.1 分析前面的测试](#)

[10.3.2 优化](#)

[10.3.3 也可以保存订单](#)

[10.3.4 结论](#)

[第十一章 多对多映射](#)

[11.1 多对多实现](#)

[11.1.1 创建User实体类](#)

[11.1.2 创建Role实体类](#)

[11.1.3 User映射配置](#)

[11.1.4 Role配置](#)

[11.1.5 核心配置文件添加映射路径](#)

[11.1.6 测试增加](#)

[11.2 级联保存](#)

[11.3 级联删除](#)

[第十二章 一对一映射](#)

[12.1 一对一外键关联](#)

[12.1.1 创建持久化类](#)

[12.1.2 配置映射文件](#)

[12.1.3 核心配置](#)

[12.1.4 测试](#)

[12.2 一对一主键关联](#)

[12.2.1 创建持久化类](#)

[12.2.2 配置](#)

[12.2.3 修改核心配置文件](#)

[12.2.4 测试](#)

课前默写

1. Hibernate中事务传播性的几种方案
2. Hibernate中的主键的几种实现策略
3. Hibernate对象的三种状态相互转换的方法

课程回顾

1. Hibernate中事务特征及其传播性
2. 更新数据丢失问题
3. Hibernate中的主键策略
4. Hibernate对象的三种状态
5. Hibernate的一级缓存策略

今天任务

1. 一对多映射
2. 多对多映射
3. 一对一映射

教学目标

1. 掌握一对多映射
2. 掌握多对多映射
3. 掌握一对一映射

第十章 一对多映射

10.1 一对多实现

10.1.1 准备项目

- 创建项目：`hibernate-02-relation`
- 引入jar，同前一个项目
- 复制实体（客户）、映射、配置、工具类

10.1.2 创建订单表

表名: t_order

语句

```

CREATE TABLE `t_order` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT 'id',
  `orderno` varchar(20) DEFAULT NULL COMMENT '订单编号',
  `product_name` varchar(100) DEFAULT NULL COMMENT '商品名称',
  `customer_id` bigint(20) DEFAULT NULL COMMENT '客户id',
  PRIMARY KEY (`id`),
  KEY `order_customer_fk` (`customer_id`),
  CONSTRAINT `order_customer_fk` FOREIGN KEY (`customer_id`) REFERENCES
`t_customer` (`c_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8

```

10.1.3 创建Order实体类

```

/**
 * 订单（多方）
 */
public class Order {
    private Long id;
    private String orderno;
    private String productName;

    //关联客户
    private Customer customer;

    //getter setter toString
}

```

10.1.4 修改Customer实体类

添加关联订单

```

/**
 * 客户（一方）
 */
public class Customer{
    private Long id;
    private String name;
    private Character gender;
    private Integer age;
    private String level;

    //关联订单
    private Set<Order> orders = new HashSet<Order>();
    //getter setter toString
}

```

10.1.5 Customer配置一对多

```

<class name="Customer" table="t_customer">
    .....
    <!-- 一对多配置 -->
    <set name="orders">
        <!-- 外键字段名称 -->
        <key column="customer_id"></key>
        <one-to-many class="Order"/>
    </set>
</class>

```

10.1.6 Order配置多对一

```

<hibernate-mapping package="com.qfedu.hibernate.pojo.one2many">

    <class name="Order" table="t_order">
        <id name="id" column="id">
            <generator class="native"></generator>
        </id>
        <property name="orderno" column="orderno"></property>
        <property name="productName" column="product_name"></property>

        <!-- 多对一配置
            name javaBean中的属性
            class 属性的全路径
            column 对应的列名
        -->
        -->
        <many-to-one name="customer" class="com.itqf.domain.Customer"
column="customer_id" />
    </class>
</hibernate-mapping>

```

10.1.7 将映射文件加入hibernate.cfg.xml

```

<mapping resource="com/qfedu/hibernate/pojo/one2many/Customer.hbm.xml"/>
<mapping resource="com/qfedu/hibernate/pojo/one2many/Order.hbm.xml"/>

```

10.1.8 测试新增关联数据

```

public class One2manyTest {
    /**
     * 需求: 1个客户 2张订单
     */
    @Test
    public void testCreateOrder(){
        //准备数据
        Customer cust = new Customer();
        cust.setName("海伦");
        cust.setGender('女');
        cust.setAge(18);
        cust.setLevel("VIP");

        Order o1 = new Order();
        o1.setOrderno("201709070001");
        o1.setProductName("JavaWeb开发详解");

        Order o2 = new Order();
        o2.setOrderno("201709070002");
        o2.setProductName("Spring开发详解");

        Session session = HibernateUtil.openSession();
        Transaction tx = session.beginTransaction();

        //建立一对多双向关系
        cust.getOrders().add(o1);
        cust.getOrders().add(o2);

        o1.setCustomer(cust);
        o2.setCustomer(cust);

        session.save(cust);
        session.save(o1);
        session.save(o2);

        tx.commit();
        session.close();
    }
}

```

10.1.9 测试查询订单

```

/**
 * 查询操作
 */
@Test
public void testSearch(){
    Session session = HibernateUtil.openSession();
    Transaction tx = session.beginTransaction();

    //查询一个客户，关联查询订单
    Customer cust = session.get(Customer.class, 3L);
    System.out.println(cust.getName()+"的订单：");
    Set<Order> orders = cust.getOrders();
    for (Order order : orders) {
        System.out.println(order.getOrderno()+" "+order.getProductName());
    }

    tx.commit();
    session.close();
}

```

10.2 cascade级联操作

10.2.1 级联保存

当只保存双向关联关系的一方时，会报告错误，此时应该在customer中配置级联保存

级联操作：就是操作一个对象的时候，想同时操作它的关联对象。

修改映射文件

```
<set name="orders" cascade="save-update">
```

如下用例，可以先测试查看报错信息；再配置上面的级联保存，然后再次进行测试，成功。

```

/**
 * 保存操作 - 级联保存
 */
@Test
public void testCascadeSave(){
    //准备数据
    Customer cust = new Customer();
    cust.setName("海伦");
    cust.setGender('女');
    cust.setAge(18);
    cust.setLevel("VIP");

    Order o1 = new Order();
    o1.setOrderno("201709070001");
    o1.setProductName("JavaWeb开发详解");

    Order o2 = new Order();
    o2.setOrderno("201709070002");
    o2.setProductName("Spring开发详解");

    Session session = HibernateUtil.openSession();
    Transaction tx = session.beginTransaction();

    //建立一对多单向关联
    cust.getOrders().add(o1);
    cust.getOrders().add(o2);
    //o1.setCustomer(cust);
    //o2.setCustomer(cust);

    session.save(cust); //使用级联保存 ( 想保存客户的时候, 同时保存订单 )
    //session.save(o1); //设置级联保存后不用保存订单
    //session.save(o2);

    tx.commit();
    session.close();
}

```

10.2.2 级联删除

当只删除父记录时, 在删除客户的时候, Hibernate会把订单表的外键值置空, 此时可以配置级联删除


```
<set name="orders" cascade="save-update,delete">
```

测试代码

```
/**
 * 级联删除
 * 注意：
 * 1) 如果没有级联删除，那么在删除客户的时候，会把订单表的cust_id外键值设置为null
 * 2) 有了级联删除，那么在删除客户的时候，会同时把该客户的所有订单删除
 */
@Test
public void testCascadeDelete(){
    //准备数据

    Session session = HibernateUtil.openSession();
    Transaction tx = session.beginTransaction();

    Customer cust = session.get(Customer.class, 4L);
    session.delete(cust);

    tx.commit();
    session.close();
}
```

10.3 inverse关系反转

10.3.1 分析前面的测试

1. 运行级联保存的测试用例
2. 查看日志中的sql语句

插入一个用户、两个订单，应该执行3个insert语句

但是发现日志中多打印了两个update语句

默认情况下inverse的值是false：

```
<set name="orders" cascade="all" inverse="false">
```

表示customer 一方需要维护关联关系，因此需要维护外键，有关联记录生成时，会做外键的更新操作。

而这个更新操作是没有必要的，因为order插入的时候已经将外键值插入。

所以customer中的update的语句是多余的

10.3.2 优化

inverse 配置：表示是否把关联关系的维护权反转（放弃）

false：默认值，不反转（不放弃）

true：反转（放弃）

放弃customer方的外键维护

```
<set name="orders" cascade="all" inverse="true">
```

重新测试，发现只有三条insert语句！

10.3.3 也可以保存订单

step1：保存时，保存订单

```
//建立一对多单向关联
//cust.getOrders().add(o1);
//cust.getOrders().add(o2);
o1.setCustomer(cust);
o2.setCustomer(cust);
//session.save(cust);//使用级联保存 （ 想保存客户的时候，同时向保存订单 ）
session.save(o1);
session.save(o2);
```

step2：在订单端设置级联保存

```
<!-- 多对一配置 -->
<many-to-one name="customer" class="Customer" column="customer_id"
cascade="all"/>
```

10.3.4 结论

通常在一对多的关联配置中，多方无法放弃关系维护权，所以应该放弃 1 方的维护权，意味着在 1 方加上 `inverse=true` 配置

第十一章 多对多映射

需求：用户与角色是多对多的关系

11.1 多对多实现

11.1.1 创建User实体类

```
public class User{

    private Integer id;
    private String name;

    //关联角色
    private Set<Role> roles = new HashSet<Role>();
}
```

11.1.2 创建Role实体类

```
public class Role{
    private Integer id;
    private String name;

    //关联用户
    private Set<User> users = new HashSet<User>();
}
```

11.1.3 User映射配置

```

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="com.qfedu.hibernate.pojo.many2many">

    <class name="User" table="t_user">
        <id name="id" column="id">
            <generator class="native"></generator>
        </id>
        <property name="name" column="name"></property>

        <!-- 多对多映射 -->
        <!--
            table:中间表名
        -->
        <set name="roles" table="t_user_role" >
            <!-- 当前方在中间表的外键 -->
            <key column="user_id" />
            <!-- column:对方在中间表的外键 -->
            <many-to-many class="Role" column="role_id" />
        </set>
    </class>
</hibernate-mapping>

```

11.1.4 Role配置

```

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="com.qfedu.hibernate.pojo.many2many">

    <class name="Role" table="t_role">
        <id name="id" column="id">
            <generator class="native"></generator>
        </id>
        <property name="name" column="name"></property>

        <!-- 多对多映射 -->
        <!--
            table:中间表名
        -->
        <set name="users" table="t_user_role" >
            <!-- 当前方在中间表的外键 -->
            <key column="role_id" />
            <!-- column:对方在中间表的外键 -->
            <many-to-many class="User" column="user_id" />
        </set>
    </class>

</hibernate-mapping>

```

11.1.5 核心配置文件添加映射路径

```

<mapping resource="com/qfedu/hibernate/pojo/many2many/User.hbm.xml" />
<mapping resource="com/qfedu/hibernate/pojo/many2many/Role.hbm.xml" />

```

11.1.6 测试增加

注意：以下测试用例如果直接执行，会报告联合主键插入重复的错误。因此可以在任意一方设置 inverse选项=true

```

<set name="users" table="t_user_role" inverse="true">

```

测试代码:

```

public class Many2manyTest {
    /**
     * 需求: 创建一个用户一个角色
     */
    @Test
    public void testCreateUser() {

        User u1 = new User();
        u1.setName("Helen1");

        Role r1 = new Role();
        r1.setName("超级管理员1");

        u1.getRoles().add(r1);
        r1.getUsers().add(u1);

        Session session = HibernateUtil.openSession();
        Transaction tx = session.beginTransaction();
        //双向都保存
        session.save(u1);
        session.save(r1);

        tx.commit();
        session.close();
    }
}

```

11.2 级联保存

注意：在多对多的保存中，如果不设置级联保存，也不设置inverse="true",那么会报告联合主键重复的错误。

可以设置级联保存，在User的多对多关联中设置如下：

```

<set name="roles" table="t_user_role" cascade="save-update">

```

测试代码:

```

public class Many2manyTest {
    /**
     * 需求: 创建一个用户一个角色
     */
    @Test
    public void testCreateUser() {

        User u1 = new User();
        u1.setName("Helen1");

        Role r1 = new Role();
        r1.setName("超级管理员1");

        u1.getRoles().add(r1);
        //r1.getUsers().add(u1);

        Session session = HibernateUtil.openSession();
        Transaction tx = session.beginTransaction();

        session.save(u1);
        //session.save(r1);

        tx.commit();
        session.close();
    }
}

```

11.3 级联删除

当没有设置级联删除的时候，如果删除User表中的记录，那么只删除User表和关联表中的记录

当设置了级联删除的时候，如果删除User表中的记录，那么会将User表、关联表和Role表中的记录全部删除！

```

<set name="roles" table="t_user_role" cascade="save-update,delete">

```

测试:

```

@Test
public void testCascadeDelete() {

    Session session = HibernateUtil.openSession();
    Transaction tx = session.beginTransaction();

    User u = session.get(User.class, 6);
    session.delete(u);

    tx.commit();
    session.close();
}

```

第十二章 一对一映射

需求：公民表和身份证表是一对一的关系

设计表的两种方案：

方案一：外键关联方案

公民	t_person
id	name
1	helen
2	tom

身份证	t_card	
id	cardno	person_id (fk+unique)
8	1234	1
9	4321	2

方案二：主键关联

公民	t_person
id	name
1	helen
2	tom

身份证	t_card
id (pk+fk)	cardno
1	1234
2	4321

12.1 一对一外键关联

12.1.1 创建持久化类

Person

```

package com.qfedu.hibernate.pojo.one2one_fk;
public class Person {
    private Integer id;
    private String name;

    //关联身份证
    private Card card;
}

```


Card

```
package com.qfedu.hibernate.pojo.one2one_fk;
public class Card {
    private Integer id;
    private String cardno;

    //关联公民
    private Person person;
}
```

12.1.2 配置映射文件

Person.hbm.xml

```
<hibernate-mapping package="com.qfedu.hibernate.pojo.one2one_fk">

    <class name="Person" table="t_person">
        <id name="id" column="id">
            <generator class="native"></generator>
        </id>
        <property name="name" column="name"></property>

        <!-- 一对一映射 -->
        <one-to-one name="card" class="Card" />
    </class>

</hibernate-mapping>
```

Card.hbm.xml

```
<hibernate-mapping package="com.qfedu.hibernate.pojo.one2one_fk">

    <class name="Card" table="t_card">
        <id name="id" column="id">
            <generator class="native"></generator>
        </id>
        <property name="cardno" column="cardno"></property>

        <!-- 唯一外键 (一对一) -->
        <many-to-one name="person" class="Person" column="person_id"
unique="true" />
    </class>

</hibernate-mapping>
```

12.1.3 核心配置

```
<mapping resource="com/qfedu/hibernate/pojo/one2one_fk/Person.hbm.xml"/>
<mapping resource="com/qfedu/hibernate/pojo/one2one_fk/Card.hbm.xml"/>
```

12.1.4 测试

```

public class One2OneTest {
    @Test
    public void testCreatePerson() {

        Session session = HibernateUtil.openSession();
        Transaction tx = session.beginTransaction();

        Person p = new Person();
        p.setName("Helen");

        Card c = new Card();
        c.setCardno("1234");

        p.setCard(c);
        c.setPerson(p);

        session.save(p);
        session.save(c);

        tx.commit();
        session.close();
    }
}

```

12.2 一对一主键关联

12.2.1 创建持久化类

Person

```

public class Person {
    private Integer id;
    private String name;

    //关联身份证
    private Card card;
}

```

Card

```
package com.qfedu.hibernate.pojo.one2one_pk;
public class Card {
    private Integer id;
    private String cardno;

    //关联公民
    private Person person;
}
```

12.2.2 配置

Person.hbm.xml

```
<hibernate-mapping package="com.qfedu.hibernate.pojo.one2one_pk">

    <class name="Person" table="t_person_pk">
        <id name="id" column="id">
            <generator class="native"></generator>
        </id>
        <property name="name" column="name"></property>

        <!-- 主键（一对一映射） -->
        <one-to-one name="card" class="Card" />
    </class>

</hibernate-mapping>
```

Card.hbm.xml

```
<hibernate-mapping package="com.qfedu.hibernate.pojo.one2one_pk">

    <class name="Card" table="t_card_pk">
        <id name="id" column="id">
            <generator class="native"></generator>
        </id>
        <property name="cardno" column="cardno"></property>

        <!-- 关联主键 (一对一) -->
        <one-to-one name="person" class="Person" constrained="true" />

    </class>

</hibernate-mapping>
```

12.2.3 修改核心配置文件

```
<mapping resource="com/qfedu/hibernate/pojo/one2one_pk/Person.hbm.xml" />
<mapping resource="com/qfedu/hibernate/pojo/one2one_pk/Card.hbm.xml" />
```

12.2.4 测试

```
public class One2OneTestPK {  
    @Test  
    public void testCreatePerson() {  
  
        Session session = HibernateUtil.openSession();  
        Transaction tx = session.beginTransaction();  
  
        Person p = new Person();  
        p.setName("Helen");  
  
        Card c = new Card();  
        c.setCardno("1234");  
  
        p.setCard(c);  
        c.setPerson(p);  
  
        session.save(p);  
        session.save(c);  
  
        tx.commit();  
        session.close();  
    }  
}
```

总结

作业

一、使用如下表关系完成以下功能

1. 配置表之间的关系
2. 使用单元测试完成各表的CRUD操作（测试级联操作）

表名	employee	中文表名称	人员信息表		
序号	字段名称	字段说明	类型	属性	备注
1	id	编号	int	自增	主键
2	name	用户名	varchar(20)	非空	
3	birthday	生日	varchar(20)	非空	
4	gender	性别	varchar(4)		
5	career	职业	varchar(20)		
6	address	地址	varchar(50)		
7	mobile	手机	varchar(20)		
8	posid	职位	int		外键

表名	position	中文表名称	职业信息表		
序号	字段名称	字段说明	类型	属性	备注
1	id	编号	int	自增	主键
2	name	职业名称	varchar(20)	非空	

二、根据以下关系自己设计表，并完成各表的CRUD操作

用户对角色（多对一）

角色对权限（多对多）

提示：一共建立4张表

面试题

1. Hibernate中关联映射的配置方法
2. Hibernate多对多映射方案（组合主键、非组合主键）
3. Hibernate中Cascade关键字和inverse关键字的用法