Hibernate查询与检索策略

一、对象导航查询、OID查询与本地SQL查询

1.1 对象导航图查询

对象图导航查询方式根据已经加载的对象，导航到他的关联对象。他是利用类与类之间的关系来检索对象的。
前面我们已经介绍过一对多，多对多的关系创建表原则。除了在实体类中创建相应的另一个实体类的对象（在多的一方：单纯一个另外实体类的对象；在"一"的一方：另外一个实体类对象的集合，且需要生成各自的get和set方法）之外，还要在各自的映射文件的set标签中中配置one-to-many(在"一"的一方)和many-to-one(在"多"的一方)。这样的话，我们只需获取某一个实体类的对象，调用其对应于另一实体类的get方法就能拿到。
下图就是获得某一方的对象(根据id查询)，再调用相应的get方法获取另一方对象即可。

```
30              Customer customer = session.get(Customer.class, 1);
31              Set<Contacter> contacters = customer.getSetContacter();
32              for (Contacter contacter : contacters) {
33                 System.out.println(contacter);
34              }
```

```
Contacter [con_id=7, con_name=季某, con_gender=女, con_phone=1355656565]
Contacter [con_id=3, con_name=张某, con_gender=男, con_phone=1233434334]
Contacter [con_id=4, con_name=陈某, con_gender=女, con_phone=1234534901]
```

1.2 OID(Object ID)查询

OID查询指的是用Session的get()和load()方法加载某条记录对应的对象。就是上面的第一句代码。

1.3 本地SQL查询

可能有的应用程序可能需要根据底层数据库的SQL方言

```
SQLQuery sqlQuery=session.createSQLQuery("select id，name，age，city from cust
omer");
```

二、HQL(Hibernate Query Language)查询

HQL(Hibernate Query Language)是而向对象的查询语言，它和SQL查询语言有些相似，但它使用的是类、对象和属性的概念，而没有表和字段的概念。在Hibernate提供的各种检索方式中，HQL是官方推荐的查询语言，也是使用最广泛的一种检索方式。它通过操作Query对象来操作hql语句。

```
Query query=session.createQuery("hql语句");
```

2.1 hql查询所有

```
//1.创建query对象所有
Query query=session.createQuery("from Customer");
//2.调用方法得到结果
List<Customer> list=query.list();
for(Customer customer:list){
    System.out.println(customer);
}
```

2.2 hql条件查询

```
//1.创建query对象进行条件查询
Query queryOptional=session.createQuery("from Customer  where cid=?");
//Query queryOptional=session.createQuery("from Customer c where c.cid=?");
//2.设置占位符号的值
//第一个参数：?的位置，注意与jdbc不同，这里的问号从0开始
//第二个参数：具体参数值
queryOptional.setParameter(0, 1);
//3.调用方法得到结果
List<Customer> listOptional=queryOptional.list();
for(Customer customer:listOptional){
    System.out.println(customer);
}
```

2.3 hql模糊查询

```
//1.创建query对象进行模糊查询
Query queryMohu=session.createQuery("from Customer c where c.custName like ?");
//2.设置占位符号的值
//第一个参数：?的位置，注意与jdbc不同，这里的问号从0开始
//第二个参数：具体参数值
queryMohu.setParameter(0, "%理%");
//3.调用方法得到结果
List<Customer> listMohu=queryMohu.list();
for(Customer customer:listMohu){
    System.out.println(customer);
}
```

2.4 hql排序查询

```
//1.创建query对象进行排序查询
Query queryOrder=session.createQuery("from Customer order by cid asc");

//2.调用方法得到结果
List<Customer> listOrder=queryOrder.list();
for(Customer customer:listOrder){
    System.out.println(customer);
}
```

2.5 hql分页查询

```
77              //1.创建query对象进行分页查询
78          Query queryPage=session.createQuery("from Customer");
79              //2.设置分页的参数
80              //2.1设置开始位置
81          queryPage.setFirstResult(0);
82              //2.2设置每页最大记录数
83          queryPage.setMaxResults(2);
84
85          //3.调用方法得到结果
86          List<Customer> listPage=queryPage.list();
87          for(Customer customer:listPage){
88              System.out.println(customer);
89          }
```

2.6 hql投影查询(只要实体类的某几个属性而已)

```
91           //1.投影查询，只查部分字段值
92          Query queryTouying=session.createQuery("select custName from Customer");
93          //2.调用方法得到结果
94          List<Object> listTouying=queryTouying.list();
95          for(Object customer:listTouying){
96              System.out.println(customer);
97          }
```

2.7 hql聚合函数查询

```
100             //1.聚合函数
101             Query queryJuhe=session.createQuery("select count(*) from Customer");
102             //2.调用方法得到结果
103             //System.out.println(queryJuhe.uniqueResult());
104             Object obj = queryJuhe.uniqueResult();
105             Long lobj=(Long)obj;
106             int count=lobj.intValue();
107             System.out.println(count);
```

三、QBC(Query By Criteria)查询

不需要写sql语句或hql语句 创建Criteria对象： `Criteria criteria = session.createCriteria(实体类名.class);`
创建条件对象：`Criterion criterion=Restrictions.静态方法`
添加查询条件：`criteria.add(条件对象);`

3.1 QBC查询所有

```
32              //1.QBC查询所有
33              Criteria criteria = session.createCriteria(Customer.class);
34              //2.调用方法得到结果
35              List<Customer> list = criteria.list();
36              for (Customer customer : list) {
37                  System.out.println(customer);
38
39              }
```

3.2 QBC条件查询与模糊查询

```
39              //1.QBC条件及模糊条件查询
40              Criteria criteria = session.createCriteria(Customer.class);
41              //2.设置条件值
42              //2.1首先使用add方法
43              //2.2使用Restrictions类设置条件
44  //          criteria.add(Restrictions.eq("cid",2));
45  //          criteria.add(Restrictions.eq("custName","中山大学"));
46              criteria.add(Restrictions.like("custName","%山大%"));
47              //3.调用方法得到结果
48              List<Customer> list = criteria.list();
49              for (Customer customer : list) {
50                  System.out.println(customer);
51
52              }
```

3.3 QBC排序查询

```
54              //1.QBC排序查询
55                Criteria criteria = session.createCriteria(Customer.class);
56              //2.设置按哪个属性进行排序desc降序，asc升序
57                criteria.addOrder(Order.desc("custLevel"));
58                //3.调用方法得到结果
59                List<Customer> list = criteria.list();
60                for (Customer customer : list) {
61                    System.out.println(customer);
62
63                }
```

3.4 QBC分页查询

```
65              //1.QBC分页查询
66              Criteria criteria = session.createCriteria(Customer.class);
67              //2.设置分页参数
68              criteria.setFirstResult(0);
69              criteria.setMaxResults(3);
70              //3.调用方法得到结果
71              List<Customer> list = criteria.list();
72              for (Customer customer : list) {
73              System.out.println(customer);
74
75              }
```

3.5 QBC统计查询

```java
//1.QBC统计查询
Criteria criteria = session.createCriteria(Customer.class);
//2.
criteria.setProjection(Projections.rowCount());
//3.
Object obj = criteria.uniqueResult();
Long lobj=(Long)obj;
int count=lobj.intValue();
System.out.println(count);
```

3.6 QBC离线查询

DetachedCriteria翻译为离线条件查询，因为它是可以脱离Session来使用的一种条件查询对象，我们都知道Criteria对象必须通过Session对象来创建。那么也就是说**必须先有Session才可以生成Criteria对象**。而**DetachedCriteria对象可以在其他层对条件进行封装。**
这个对象也是比较有用的，尤其在SSH介绍以后这个对象经常会使用。它的主要优点是做一些特别复杂的条件查询的时候，往往会**在WEB层向service层传递很多的参数，业务层又会将这些参数传递给DAO层，最后在DAO中拼接SQL完成查询。**有了离线条件查询对象后，那么这些工作都可以不用关心了，我们可以在WEB层将数据封装好，传递到业务层，再由业务层传递给DAO完成。

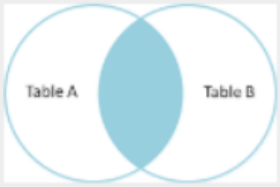
```java
//1.QBC离线查询，与session无关
DetachedCriteria detachedCriteria=DetachedCriteria.forClass(Customer.class);
//2.最后执行时候才需要session
Criteria criteria=detachedCriteria.getExecutableCriteria(session);
//3.设置分页参数
criteria.setFirstResult(0);
criteria.setMaxResults(3);
//4.调用方法得到结果
List<Customer> list = criteria.list();
for (Customer customer : list) {
System.out.println(customer);

}
```

四、HQL多表查询

4.1 SQL多表查询回顾

4.1.1 内连接查询(求交集)



- 隐式内联合查询

```
SELECT * FROM person,dept WHERE person.dept_id = dept.did;
```

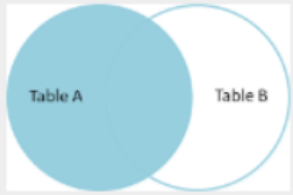
- 显式内联合查询

```
-- 显式内联合查询 innner可以省略
SELECT * FROM person INNER JOIN dept  ON person.dept_id = dept.did;

SELECT * FROM person JOIN dept  ON person.dept_id = dept.did;
```

4.1.2 外连接查询(优先显示某个集合)

- 左外联合查询 left outer join 其中outer可以省略 (左边表中的数据优先全部显示)




```
SELECT * FROM person LEFT JOIN dept  ON person.dept_id = dept.did;
```

- 右外联合查询 right outer join 其中outer可以省略 (右边表中的数据优先全部显示)

```
SELECT * FROM person RIGHT JOIN dept  ON person.dept_id = dept.did;
```

4.2 HQL多表查询

4.2.1 内连接与迫切内连接

```
28              //1.内连接查询
29              Query query=session.createQuery("from Customer c inner join c.setContacter");
30              List list=query.list();
```

```
32              //2.迫切内连接，返回是对象，而不是数组而已与内连接一样
33              Query query=session.createQuery("from Customer c inner join fetch c.setContacter");
34              List list = query.list();
```

4.2.2 左外连接与迫切左外连接

```
//1.左外连接查询
Query query=session.createQuery("from Customer c left outer join c.setContacter");
List list=query.list();
```

```
//2.迫切外连接，返回是对象，而不是数组而已与内连接一样
Query query2=session.createQuery("from Customer c left outer join fetch c.setContacter");
List list2 = query2.list();
```

4.2.3 右外连接(注意没有迫切右外连接)

略

五、Hibernate检索策略

5.1 立即查询(调用get方法)

```
//执行下面这行代码是否会立即执行sql语句来分辨是否是立即查询
Customer scut=session.get(Customer.class, 4);
//直接用其set集合遍历即可
Set<Contacter> contacters = scut.getSetContacter();
for(Contacter contacter:contacters)
```

Console Tasks

HibernateQueryWays [JUnit] D:\runners\JavaJDK\JRE7\bin\javaw.exe (2019年5月12日 下午8:55:25)

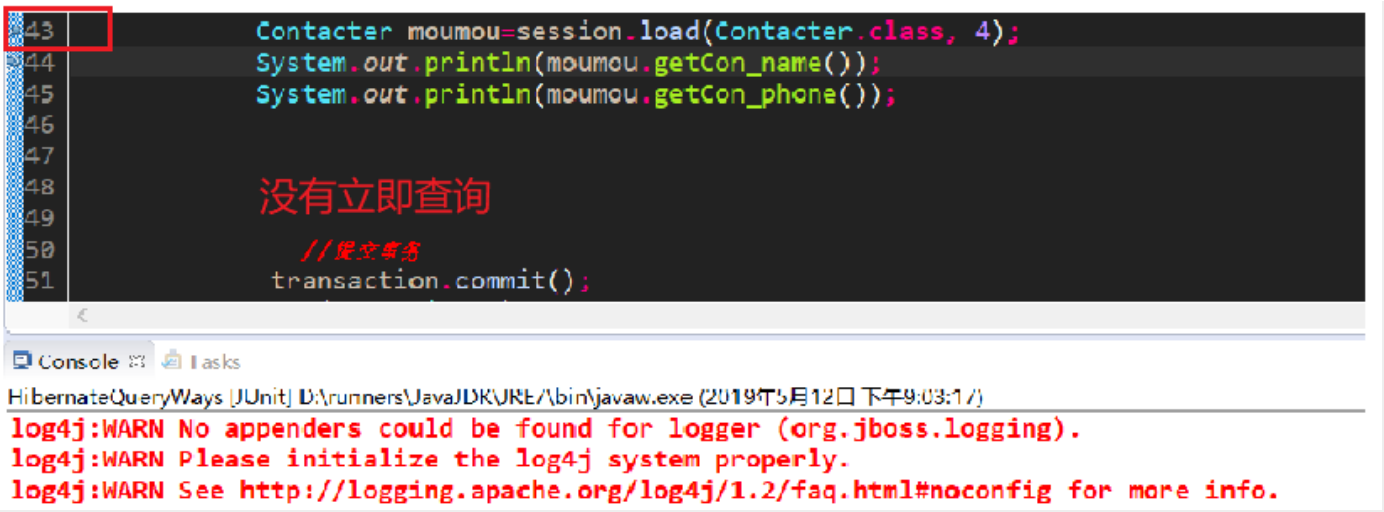
```
log4j:WARN No appenders could be found for logger (org.jboss.logging).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Hibernate:
    select
        customer0_.cid as cid1_1_0_,
        customer0_.custName as custName2_1_0_,
        customer0_.custLevel as custLeve3_1_0_,
        customer0_.custSource as custSour4_1_0_,
        customer0_.custPhone as custPhon5_1_0_,
        customer0_.custMobile as custMobi6_1_0_
    from
        Customer customer0_
    where
        customer0_.cid=?
```

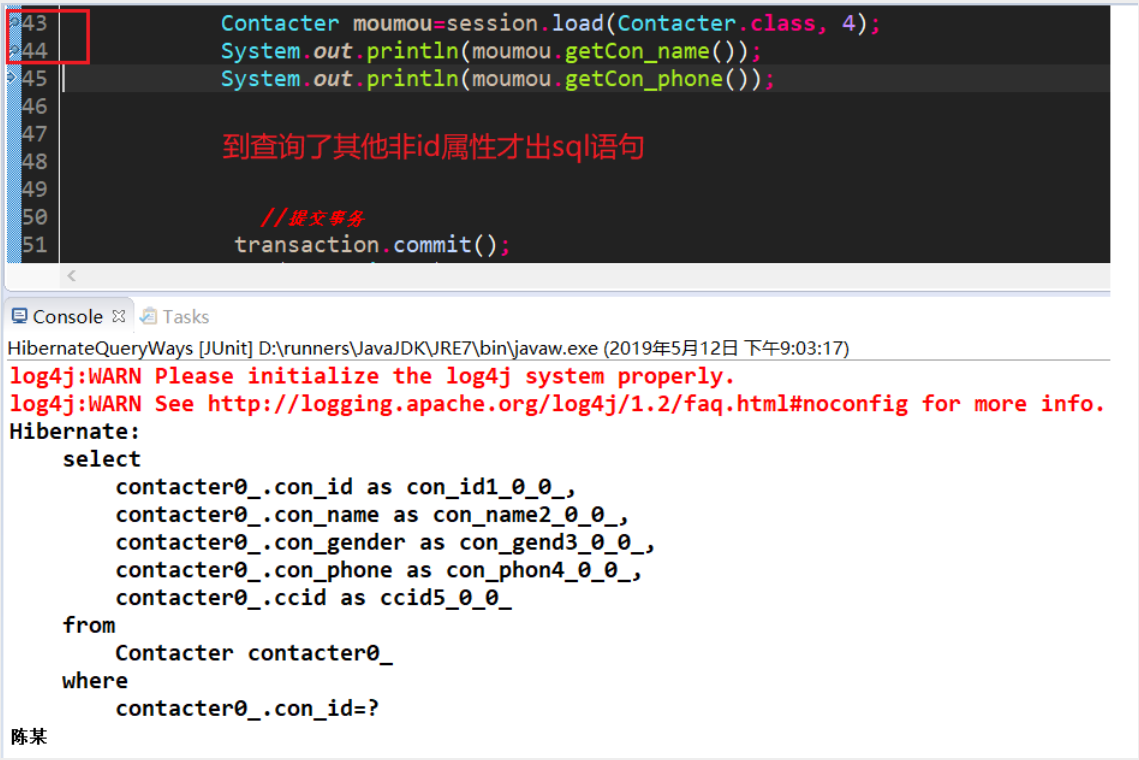
5.2 延迟查询

5.2.1 类别级别延迟(调用load方法)

根据id 查询返回实体对象，调用load方法不会立即发送语句

```
//直接用其set集合遍历即可
//不会立即查询
//返回的对象只有id值
//得到的对象里面不是id的其他值时才会发送语句
Contacter moumou=session.load(Contacter.class, 4);
System.out.println(moumou.getCon_name());
System.out.println(moumou.getCon_phone());
```





5.2.2 关联级别延迟

查询某个主表对应的实体类对象，再查询对应从表的实体类对象的过程是否需要延迟，称作关联级别延迟

```
47              //默认关联级别延迟 ：这当然会
48             Customer scut=session.get(Customer.class, 4);
49             //得到set集合，是否会发语句：不会
50              Set<Contacter> contacters = scut.getSetContacter();
51
52              //是否会发语句:会
53              System.out.println(contacters.size());
54
```

一对多和多对多检索策略: <set> 的 lazy 和 fetch 属性

fetch (默认值select)	Lazy (默认值是true)	策略
Join	false	采用迫切左外联接检索。
Join	true	采用迫切左外联接检索。
join	extra	采用迫切左外联接检索。
select	false	采用立即检索
select	true	采用延迟检索
select	extra	采用延迟检索（及其懒惰）
subselect	false/true/extra 也分为3中情况	嵌套子查询(检索多个customer对象时) Lazy属性决定检索策略)

六、Hibernate批量抓取