http://blog.csdn.net/damogu_arthur/article/details/46924805

一、一对多

以班级Classes和学生Student为例：

回忆sql语句:

```
//内链接,两种方式效果一样,查询的是两边都有的数据
SELECT c.*,s.* FROM classes c,student s WHERE s.cid=c.cid;

SELECT c.cname,s.sname FROM classes c INNER JOIN student s ON
s.cid=c.cid;

//左外连接，在内链接基础上，左边表有而右边表没有，两种方式等效;
SELECT c.* ,s.* FROM student s LEFT OUTER JOIN classes c ON
s.cid=c.cid;
SELECT c.* ,s.* FROM student s LEFT  JOIN classes c ON
s.cid=c.cid;

//右外连接，在内链接基础上，右边有而左边无，两种方式等效;
SELECT c.* ,s.* FROM classes c RIGHT  OUTER JOIN student s ON
s.cid=c.cid;
SELECT c.* ,s.* FROM classes c RIGHT  JOIN student s ON
s.cid=c.cid;
```

HQL语句：

```
//查询所有：
from Classes c,Student s where c.cid=s.classes.cid;
//选择某些属性查询
```

select c.cname,s.sname from Classes c,Student s where c.cid=s.classes.cid;

//选择某些属性，封装为bean查询；

```
select new
cn.itheima03.hibernate.domain.ClassesView(c.cname,s.sname)    from
Classes c,Student s where c.cid=s.classes.cid;
```

//内链接查询，得到的是两个bean

```
from Classes c inner join c.students s;
```

//内敛链接查询，得到的是Classes对象，对象中包含studet集合

```
from Classes c inner join fetch c.students s;

from Student s inner join fetch s.classes c;

select new
cn.itheima03.hibernate.domain.ClassesView(c.cname,s.sname)    from
 Student s inner join   s.classes c ;

from Classes c left outer join fetch c.students s;

from Student s left outer join fetch s.classes;
```

示例代码：

```
/**
 * 1.一对多
 * sql:select c.*,s.* from classes c,student s where c.cid=s.cid;
 * hql:from Classes c,Student s where c.cid=s.classes.cid，注意与上句的区别；
 * 得到的list是object[]，数组中的元素是Classes和Student对象；
 *
 */
 @Test
 public void testOneToMany_EQ(){
 Session session = sessionFactory.openSession();
 Query query = session.createQuery( "from Classes c,Student s where 
c.cid=s.classes.cid");
  List list = query.list();
 System. out.println(query.list().size());
 session.close();
 }

 /**
 * 2.带属性的查询；
 * list中装的是object[];
 */
 @Test
 public void testOneToMany_EQ_Property(){
 Session session = sessionFactory.openSession();
 Query query = session.createQuery( "select c.cname,s.sname from Classes 
c,Student s where c.cid=s.classes.cid");
 query.list();
 session.close();
 }
 /**
 * 3.带属性查询，将查询结果封装成一个bean；
 * 得到的list中装的是classView对象；
 */
 @Test
 public void testOneToMany_EQ_Property_Constructor(){
 Session session = sessionFactory.openSession();
 Query query = session.createQuery( "select new 
cn.itheima03.hibernate.domain.ClassesView(c.cname,s.sname) " +
  "from Classes c,Student s where c.cid=s.classes.cid");
  List list = query.list();

 session.close();
 }

 /**
 * 4.内连接
 * 结果与例子1一样；
 */
 @Test
 public void testOneToMany_InnerJoin_Query(){
 Session session = sessionFactory.openSession();
 StringBuffer buffer = new StringBuffer();
 buffer.append( "from Classes c inner join c.students s");
```

```
Query query = session.createQuery(buffer.toString());
query.list();
session.close();
}

/**
* 5.迫切内连接1：获取所有有学生的班级及班级下的学生；
* 要想得到的集合中装的Classes对象，对象中set集合中装student，可以使用迫切内
接。
*
*/
@Test
public void testOneToMany_InnerJoin_Fetch_Query_1(){
Session session = sessionFactory.openSession();
StringBuffer buffer = new StringBuffer();
buffer.append( "from Classes c inner join fetch c.students s");
Query query = session.createQuery(buffer.toString());
 List list = query.list();
session.close();
}

/**
* 6.迫切内连接2
* 从学生端出发；
*/
@Test
public void testOneToMany_InnerJoin_Fetch_Query_2(){
Session session = sessionFactory.openSession();
StringBuffer buffer = new StringBuffer();
buffer.append( "from Student s inner join fetch s.classes c");
Query query = session.createQuery(buffer.toString());
query.list();
session.close();
}

/**
* 7.迫切内连接3：获取属性，封装结果；
* select new cn.itheima03.hibernate.domain.ClassView(c.cname,s.sname)
* from Student s inner join fetch s.classes c；
* 上述的 hql语句会报错，因为from后面想要的结构和select想要的结构是冲突的，
如果在from后面加fetch,不能写select语句，如果加select，不能写fetch,两者只能选
一
*
```

```
*/
@Test
public void testOneToMany_InnerJoin_Fetch_Query_Property(){
Session session = sessionFactory.openSession();
StringBuffer buffer = new StringBuffer();
 //下面的写法不对；
// buffer.append("select new
cn.itheima03.hibernate.domain.ClassView(c.cname,s.sname) " +
// " from Student s inner join fetch s.classes c");

 //不要fetch;
buffer.append( "select new
cn.itheima03.hibernate.domain.ClassesView(c.cname,s.sname) " +
 " from Student s inner join s.classes c " );

Query query = session.createQuery(buffer.toString());
 List list = query.list();
session.close();
}

/**
* 8.迫切左外连接
* 从班级出发，得到班级对应的学生
*/
@Test
public void testOneToMany_LeftJoin_Fetch(){
Session session = sessionFactory.openSession();
StringBuffer buffer = new StringBuffer();
buffer.append( "from Classes c left outer join fetch c.students s");
Query query = session.createQuery(buffer.toString());
List<Classes> list = query.list();

 for (Classes classes : list) {
System. out.println("classes:" +classes.getCname());
Set<Student> students = classes.getStudents();
 for (Student student : students) {
System. out.println(" student:" +student.getSname());
}
}

session.close();
}
```

```
/**
* 9.迫切左外连接2
* 从学生出发，得到对应的班级
*/
@Test
public void testOneToMany_RightJoin_Fetch(){
Session session = sessionFactory.openSession();
StringBuffer buffer = new StringBuffer();
buffer.append( "from Student s left outer join fetch s.classes ");
Query query = session.createQuery(buffer.toString());
List<Student> list = query.list();

 for (Student student : list) {
System. out.println("student:" +student.getSname());
 if (student.getClasses()!=null) {
System. out.println(" " +student.getClasses().getCname());
}
}

session.close();
}
```

二。多对多

学生Student和课程Course为例:

Student里有装Course的set集合，Course里也有装Student的set集合;

多对多与一对多操作差不多

```
/**
 * 1.得到所有的学生以及其对应的课程
 * 从学生端出发
 * list装的是学生；
 */
@Test
public void testManyToMany_LeftJoin_Fecth(){
Session session = sessionFactory.openSession();
StringBuffer buffer = new StringBuffer();
buffer.append( "from Student s left outer join fetch s.courses");
Query query = session.createQuery(buffer.toString());
 List list = query.list();
```

```
session.close();
}
/**
* 2.得到所有的课程及课程下对应的学生；
* list装的是课程
*/
@Test
public void testManyToMany_LeftJoin_Fecth_2(){
Session session = sessionFactory.openSession();
StringBuffer buffer = new StringBuffer();
buffer.append( "from Course c left outer join fetch c.students s");
Query query = session.createQuery(buffer.toString());
query.list();
session.close();
}

/**
* 3.一对多和多对多的结合
* 得到所有班级下的所有学生以及所有学生下的所有课程；
* 从班级出发
*/
@Test
public void testManyToManyAndOneToMany(){
Session session = sessionFactory.openSession();
StringBuffer buffer = new StringBuffer();
buffer.append( "from Classes c left outer join fetch" +
 " c.students s left outer join fetch s.courses");

Query query = session.createQuery(buffer.toString());
List<Classes> classeList = query.list();
 //去掉集合中的重复元素
Set<Classes> sets = new HashSet<Classes>(classeList);
classeList = new ArrayList<Classes>(sets);

System. out.println(classeList.size());
 for(Classes classes:classeList){//遍历班级
System. out.println(classes.getCname());
Set<Student> students = classes.getStudents();//得到班级下的学生
 for(Student student:students){//遍历学生
System. out.println(student.getSname());
Set<Course> courses = student.getCourses();
 for(Course course:courses){//遍历学生下的课程
System. out.println(course.getCname());
}
}
}
session.close();
}
/**
* 从中间表出发，班级有学生，学生修课程，故从学生角度出发进行查询；
*/
@Test
public void testManyToManyAndOneToMany_2(){
Session session = sessionFactory.openSession();
StringBuffer buffer = new StringBuffer();
buffer.append( "from Student s left outer join fetch s.classes c
left outer join fetch s.courses cc");
Query query = session.createQuery(buffer.toString());
List<Student> studentList = query.list();
 for(Student student:studentList){
System. out.println(student.getSname());
Classes classes = student.getClasses();
System. out.println(classes.getCname());
Set<Course> courses = student.getCourses();
 for(Course course:courses){
System. out.println(course.getCname());
}
}
session.close();
}

/*******************************************************************************/
 /**
* 面向对象的查询
*/
 @Test
 public void testQueryCriteria(){
Session session = sessionFactory.openSession();
List<Classes> classesList = session.createCriteria(Classes.class).list() ;
System. out.println(classesList.size());
session.close();
}

 @Test
 public void testQueryCriteria_Where(){
Session session = sessionFactory.openSession();
```

```
 Classes classes =
(Classes)session.createCriteria(Classes.class).add(Restrictions.eq("cid" ,
1L)).uniqueResult();
 System. out.println(classes.getCname());
 session.close();
 }
}
```

总结:

无论是一对多还是多对多，hql语句中含有fetch时，得到的list装的是From 后面的对象，对象中可能有相关联对象的集合或者对象；