Hibernate HQL查询

hibernate提供了强大的查询系统，使用Hibernate有多种查询方法可以选择：可以使用Hibernate的HQL查询，也可以使用条件查询，甚至可以使用原生的SQL查询语句。其中HQL查询时Hibernate配置的功能强大的查询语句。HQL是非常有意识的被设计为完全面向对象的查询，它可以理解如继承、多态 和关联之类的概念。

# HQL查询

HQL的语法和SQL很相似，但是HQL是一种面向对象的查询语句，它的操作对象是类、实例、属性等，而SQL的操作对象 是数据表、列等数据库对象。

由于HQL是完全面向对象的查询语句，因此可以支持继承、多态等特性。

HQL查询依赖于Query类，每一个Query实例对应一个查询对象，它的执行是通过Session的createQuery()方法来获得的。执行HQL查询的步骤：

1. 获得Hibernate Session对象
2. 编写HQL语句
3. 调用Session的createQuery方法创建查询对象
4. 如果HQL语句包含参数，则调用Query的setXxx方法为参数赋值
5. 调用Query对象的list等方法返回查询结果。

|  |
| --- |
| private void query(){  Session session = HibernateUtil.getSession();  Transaction tx = session.beginTransaction();  //以HQL语句创建Query对象，执行setString方法为HQL语句的参数赋值  //Query调用list方法访问查询的全部实例  List list = session.createQuery("select distinct p from Person p where p.name=:name")  .setString("name", "chenssy").list();  //遍历查询结果  for (Iterator iterator = list.iterator();iterator.hasNext();) {  Person p = (Person) iterator.next();  System.out.println("id="+p.getId()+",age="+p.getAge());  }  session.close();  }  上面的程序先编写HQL语句后，使用Session的createQuery(hql)方法创建一个Query，Query对象使用setXxx方法为HQL语句的参数赋值，最后调用list()方法返回查询的全部结果。  在这里Query对象可以连续多次调用setXxx方法为HQL参数赋值。这是因为Hibernate Query的setXxx方法的返回值为Query本身，因此程序创建Query后，可以直接多次调用setXxx方法为HQL语句的参数赋值。  Query对象还包含如下两个方法：  setFirstResult(int firstResult)：设置返回的结果集从第几条记录开始  setMaxResult(int maxResult)：设置本次查询返回的结果数目  这两个方法用于对HQL查询实现分页控制 |

# HQL查询的from子句

|  |
| --- |
| #Hibernate中最简单的查询语句的形式如下：  from Person #From关键字后面紧跟持久化类的类名。  #大多数情况, 你需要指定一个别名, 原因是你可能需要在查询语句的其它部分引用到  From Person as p  #子句中可以同时出现多个类, 其查询结果是产生一个笛卡儿积或产生跨表的连接  From Person as p ,MyEvent as e |

# 关联与连接

当程序需要从多个数据表中获取数据时，Hibernate使用关联映射来处理底层数据表之间的连接，一旦我们提供了正确的关联映射后，当程序通过Hibernate进行持久化访问时，将可利用Hibernate的关联来进行连接。

HQL支持两种关联join的形式：implicit(隐式) 与explicit（显式）。

显式form子句中明确给出了join关键字，而隐式使用英文点号(.)来连接关联实体。受支持的连接类型是从ANSI SQL中借鉴来的。

* inner join（内连接）
* left outer join（左外连接）
* right outer join（右外连接）
* full join (全连接，并不常用)

注意：使用显式连接，可以通过with关键字来提供额外的join条件，例如：

From Person as p inner join p.myEvent e with p.id=e.id

从上面可以看出with关键字的作用等同于SQL中on关键字的作用：用于指定连接条件。还有，一个"fetch"连接允许仅仅使用一个选择语句就将相关联的对象或一组值的集合随着他们的父对象的初始化而被初始化，这种方法在使用到集合的情况下尤其有用，对于关联和集合来说，它有效的代替了映射文件中的外联接与延迟声明（lazy declarations）。

对于隐式连接和显示连接有如下两个区别：

1. 显式连接底层将转换成SQL99的交叉连接，显式连接底层将转换成SQL99的inner join、left join、right join等连接。
2. 隐式连接和显式连接查询后返回的结果不同。使用隐式连接查询返回的结果是多个被查询实体组成的集合。使用显式连接的结果分为两种：如果HQL语句中省略select关键字时，返回的结果也是集合，但集合元素是被查询持久化对象、所有被关联的持久化对象所组成的数组。如果没有省略select关键字，返回的结果同样是集合，集合中的元素是跟在select关键字后的持久化对象组成的数组。

|  |
| --- |
| Form Person as p inner join p.myEvent as e with p.id=e.id //...........1  Select p from Person as p inner join p.myEvent as e with p.id=e.id //.........2  ........1中返回的结构是有Person实体和ＭyEvent实体组成的数组集合。  而.........2 返回的结果是只有Person组成的集合。 |

对于有集合属性的。Hibernate默认采用延迟加载策略。如对于持久化类Person，有集合属性myEvent。加载Person实例时，默认是不加载myEvent的，如果session被关闭了，Person实例将会无法访问到关联的myEvent属性的。为了解决这个问题，可以再Hibernate映射文件中配置指定:lazy="false"来关闭延迟加载。或者使用join fetch:

|  |
| --- |
| From Person as p join fetch p.myEvent  一个fetch连接通常不需要被指定别名, 因为相关联的对象不应当被用在 where 子句 (或其它任何子句)中。同时，相关联的对象并不在查询的结果中直接返回，但可以通过他们的父对象来访问到他们。使用fetch关键字时需要注意以下几个问题：   1. fetch不应该与setMaxResults()和setFirstResults()共用， 2. fetch不能与独立的with条件一起使用 3. 如果在一次查询中fetch多个集合，可以查询返回的笛卡尔积 4. full join fetch和right join fetch没有任何意义 5. 对于bag映射而言，同时join fetch多个结合时可能会出现非预期结果 |

# select子句

|  |
| --- |
| #Select子句用于选择将哪些对象与属性返回到查询结果集中。当然select选择的属性必须是from后持久化类包含的属性。  Select p.name from Person as p  #select查询语句可以返回值为任何类型的属性，包括返回类型为某种组件(Component)的属性：  Select p.myEvent.title from Person as p  #select查询语句可以返回多个对象和（或）属性，存放在 Object[]队列中：  Select p,e from Person as p inner join p.myEvent as e with p.id=e.id  #Select查询语句也支持将选择出的属性存放到一个List对象中  Select new List(p.name,p.age) from Person as p  #Select查询语句还可以将选择出的属性直接封装成一个对象。  Select new ClassTest(p.id,p.name,p.age) from Person as p  但前提是ClassTest支持p.id，p.name，p.age的构造前，假如p.id的数据类型是int，p.name的数据类型是String，p.age的数据类型是int，那么ClassTest必须有如下构造器：ClassTest(int id,String name,int age)  #Select还支持给选定的表达式名别名：  Select p.name as personname from Person as p  #这种做法在与子句select new map一起使用时最有用:  Select new map(p.name as personname) from Person as p |

# 聚集函数

受支持的聚集函数如下：

avg(...), sum(...), min(...), max(...)

count(\*)

count(...), count(distinct ...), count(all...)

Select子句支持使用distinct和all关键字，此时的效果与SQL中的效果相同。

# 多态查询

Hibernate可以理解多态查询，from后跟持久化类名，不仅会查出该持久化类的全部实例还好查询出该类的全部子类的全部实例。

|  |
| --- |
| from Person as p  该查询语句不仅会查询出Person的全部实例，还会查询出Person的子类：Teacher的全部属性。  Hibernate 可以在from子句中指定任何 Java 类或接口. 查询会返回继承了该类的所有持久化子类 的实例或返回声明了该接口的所有持久化类的实例。下面的查询语句返回所有的被持久化的对象：  From java.lang.Object o |

# Where子句

|  |
| --- |
| #where子句允许你将返回的实例列表的范围缩小. 如果没有指定别名，你可以使用属性名来直接引用属性:  From Person where age < 40  #如果指派了别名，需要使用完整的属性名:  From Person as p where p.age<40  #复合属性表达式增强了where子句的功能：  From person p where p.myEvent.title like "%你是"  #该查询语句被翻译为一个含有内连接的SQL查询语句。只要没有出现集合属性，HQL语句可使用点号来隐式连接多个数据表：  From person p where p.myEvent.event.name like "%hhh";  #=运算符不仅可以被用来比较属性的值，也可以用来比较实例：  from Cat cat, Cat rival where cat.mate = rival.mate  from Cat cat, Cat rival where cat.mate = rival.mate  #特殊属性（小写）id可以用来表示一个对象的唯一的标识符  From Person p where p.id=1  From Person p where p.myEvent.id=1  第二个查询是有效的。此时不需要进行表连接，而完全使用面向对象的方式查询！  #在进行多态持久化的情况下，class关键字用来存取一个实例的鉴别之。嵌入where自己中的java类名将会被作为该类的鉴别值。  from Person as p where p.class = Teacher  在执行多态的时候，默认会选出Person及其所有子类的实例，但是上面的HQL语句，将只会选出Teacher类的实例。  当where子句的运算符只支持基本类型或者字符串时，where子句中的属性表达式必须以基本类型或者字符串结尾，不要使用组件类型属性结尾。 |

# order by 子句

查询返回的列表(list)可以按照一个返回的类或组件（components)中的任何属性（property）进行排序： From Person as p ordery by p.id

可选的asc或desc关键字指明了按照升序或降序进行排序.

# group by子句

一个返回聚集值(aggregate values)的查询可以按照一个返回的类或组件（components)中的任何属性（property）进行分组：

|  |
| --- |
| Select p.id,p.name from Person p group by p.age  #可以使用having子句对分组进行过滤  Select p.id,p.name from Person p group by p.age having p.age between 10 and 40  注意:group by子句与 order by子句中都不能包含算术表达式。也要注意Hibernate目前不会扩展group的实体,因此你不能写group by cat,除非cat的所有属性都不是聚集的。你必须明确的列出所有的非聚集属性。 |

# 子查询

对于支持子查询的数据库，Hibernate支持在查询中使用子查询。一个子查询必须被圆括号包围起来（经常是SQL聚集函数的圆括号）。 甚至相互关联的子查询（引用到外部查询中的别名的子查询）也是允许的。

|  |
| --- |
| From Person p where p.age > (select avg(p1.age) from Person p1 )  与SQL子查询相同，如果子查询是多行结果集，则应该使用多行运算符。同时HQL子查询只可以在select子句或者where子句中出现。  如果在select子查询或的列表中包含多项，则在HQL中需要使用一个元组构造符：  From Person as p where (p.name,p.age) in (select s.name,s,age from Student as s) |

# 命名查询

HQL查询还支持将查询所用的HQL语句放在配置文件中，而不是程序代码中。通过这种方式，可以大大提高程序的解耦。

在Hibernate映射文件中使用<query.../>元素来定义命名查询。使用<query.../>元素是需要指定一个name属性，该属性指定该命名查询的名字。

|  |
| --- |
| <query name="namedQuery">  <!-- 此处确定命名查询的HQL语句 -->  from Person as p  </query> |
| Session提供一个getNamedQuery(String name)方法用于获取指定命名HQL查询并且创建Query对象  public void namedQuery(){  Session session = HibernateUtil.getSession();  Transaction tx = session.beginTransaction();  //执行命名查询  List list = session.getNamedQuery("namedQuery").list();  for (Iterator iterator = list.iterator();iterator.hasNext();) {  Person p = (Person) iterator.next();  System.out.println(p.getName());  }  tx.commit();  session.close();  } |