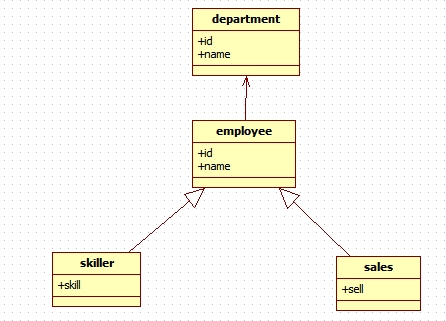
Hibernate继承映射

于面向对象的程序设计语言，继承、多态是两个最基本的概念。hibernate的继承映射可以理解两个持久化类之间的继承关系。Hibernate支持几种继承映射策略，不管哪种继承映射策略，Hibernate的多态查询都可以很好的运行。

假设有四个对象：Department、Employee、Skiller、Sales。他们四个对象存在如下关系：



从上面的关系可知：Employee与Department之间存在N-1的关联关系。Skiller和Sales都是Employee的子类。下面是这四个类的代码：

|  |
| --- |
| public class Department {  private Integer id;  private String name;  private Set<Employee> employees; //关联关系    //省略setter和getter方法  }  public class Employee {  private Integer id;  private String name;  private Department department;  //省略setter和getter方法  }  public class Skiller extends Employee {  private String skiller;  //省略setter和getter方法  }  public class Sales extends Employee {  private int sell;  //省略setter和getter方法  } |

# 采用subclass元素的继承映射

在这种继承映射策略下，整个继承树的所有实例都将保存在同一张表中。因为是将父类、子类的实例全部保存在同一张表中，所以需要在该表中添加列，通过该列可以区分每行记录到底是属于哪个类的实例---这个列被称之为辨别者。

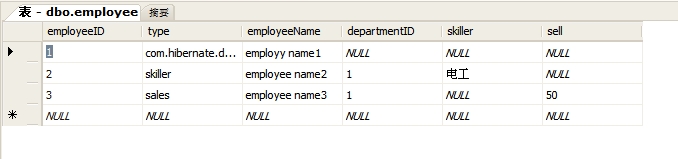
在这种继承映射策略下，我们需要使用<subclass.../>元素来映射子类的持久化类，使用<discrimainator.../>元素来映射辨别者，同时还需要给<subclass.../>元素增加discriminator-value属性来指定每个类的辨别者的值。

|  |
| --- |
| <hibernate-mapping package="com.hibernate.domain">  <class name="Employee" table="employee">  <id name="id" column="employeeID">  <generator class="hilo" />  </id>  <!-- 映射辨别者 -->  <discriminator column="type" type="string" />    <property name="name" column="employeeName" />  <!-- 用于映射N-1关联实体，指定关联实体类为 :Department,指定外键为：departmentID-->  <many-to-one name="department" column="departmentID" />    <!-- 使用subclass元素映射Employee的子类Skiller -->  <subclass name="Skiller" discriminator-value="skiller">  <property name="skiller" />  </subclass>    <!-- 使用subclass元素映射Employee的子类Sales -->  <subclass name="Sales" discriminator-value="sales">  <property name="sell" />  </subclass>  </class>  </hibernate-mapping> |

在这里我们只需要给父类进行映射就可以了。在这个配置文件中，指定了一个辨别者列：type，该列其本省是没有任何意义的，只是起到一个区分每条记录时对应哪个持久化类的作用。其中Skiller类的辨别者列type的值为skiller，Sales类的辨别者列type的值为sales。通过下面的程序段来进行操作。

|  |
| --- |
| static void add(){  Session session = HibernateUtil.getSession();  Transaction tx = session.beginTransaction();    Department department = new Department();  department.setName("department name1");    Employee emp1 = new Employee();  emp1.setName("employy name1");  emp1.setId(1);  Skiller emp2 = new Skiller();  emp2.setSkiller("电工");  emp2.setName("employee name2");  emp2.setDepartment(department); //建立对象之间的关联关系    Sales emp3= new Sales();  emp3.setSell(50);  emp3.setName("employee name3");  emp3.setDepartment(department); //建立对象之间的关联关系    session.save(department);  session.save(emp1);  session.save(emp2);  session.save(emp3);  tx.commit();  session.close();    } |
| insert into department (departmentName) values (?)  insert into employee (employeeName, departmentID, type) values (?, ?, 'com.hibernate.domain.Employee')  insert into employee (employeeName, departmentID, skiller, type) values (?, ?, ?, 'skiller')  insert into employee (employeeName, departmentID, sell, type) values (?, ?, ?, 'sales') |

在第二条sql语句，type的值为com.hibernate.domain.Employee。对于这个值可以理解为空。在第三条sql语句中，type的值为skiller，第四天sql语句中type值为sales。同时要注意第三条sql语句中的sell列是没有值的，第四条的skiller列同样也没有值。所以在这里一定要注意：使用subclass继承映射策略时，其子类中增加的属性映射的字段是一定不能有非空约束的。



通过下面的程序段，来进行查询操作

|  |
| --- |
| static void query(int empid){  Session session = HibernateUtil.getSession();  Employee emp = (Employee) session.get(Employee.class, empid);  }  Hibernate: select employee0\_.employeeID as employeeID2\_0\_, employee0\_.employeeName as employee3\_2\_0\_, employee0\_.departmentID as departme4\_2\_0\_, employee0\_.skiller as skiller2\_0\_, employee0\_.sell as sell2\_0\_, employee0\_.type as type2\_0\_  from employee employee0\_ where employee0\_.employeeID=?  从上面的sql语句可以看到它是从一张表中获取所有的记录。这正是这种继承策略的一个非常大的好处：在这种映射策略下，整棵继承树的所有数据都保存在一张表中，因此不管进行怎样的查询，不管查询继承树中的那一层的实体，底层数据库都只需要在一张表中查询即可，非常方便、高效。 |

# 采用joined-subclass元素的继承映射

采用这种策略时，父类实例保存在父类表里，而子类实例则有父类表和子类表共同存储。在这种策略下，是将父类与子类共有的属性保存在父类表中，而子类特有的属性，则保存在子类表中，就是说，父类一张表，子类一张表，同时子类表中需要增加父类表的外键。采用joined-subclass来映射这种策略，并且需要为每个子类使用<key.../>元素映射共有主键--这个主键类还将参照父表的主键列。映射文件如下：

|  |
| --- |
| <hibernate-mapping package="com.hibernate.domain">  <class name="Employee" table="employee">  <id name="id" column="employeeID">  <generator class="hilo" />  </id>    <property name="name" column="employeeName" />  <!-- 用于映射N-1关联实体，指定关联实体类为 :Department,指定外键为：departmentID-->  <many-to-one name="department" column="departmentID" />    <!-- 使用join-class元素映射Employee类的Skill而子类 -->  <joined-subclass name="Skiller">  <!-- 必须使用key元素映射父子类的共有主键 -->  <key column="employee\_id" />  <property name="skiller" />  </joined-subclass>    <!-- 使用join-class元素映射Employee类的Sales而子类 -->  <joined-subclass name="Sales">  <!-- 必须使用key元素映射父子类的共有主键 -->  <key column="employee\_id" />  <property name="sell" />  </joined-subclass>  </class> |

通过上面的增加代码，执行添加操作。得到如下结果：



|  |
| --- |
| insert into department (departmentName) values (?)  insert into employee (employeeName, departmentID) values (?, ?)  insert into employee (employeeName, departmentID) values (?, ?)  insert into Skiller (skiller, employee\_id) values (?, ?)  insert into employee (employeeName, departmentID) values (?, ?)  insert into Sales (sell, employee\_id) values (?, ?) |

从上面的SQL语句中也可以看出hibernate对与这种继承映射策略的处理。父类一张表，子类一张表同时子类引用父类外键建立关联关系。对与这样继承映射策略的查询，hibernate采用表连接方式来获取子类表的信息：

|  |
| --- |
| Hibernate: select employee0\_.employeeID as employeeID2\_0\_, employee0\_.employeeName as employee2\_2\_0\_, employee0\_.departmentID as departme3\_2\_0\_, employee0\_1\_.skiller as skiller3\_0\_, employee0\_2\_.sell as sell4\_0\_, case  when employee0\_1\_.employee\_id is not null then 1  when employee0\_2\_.employee\_id is not null then 2  when employee0\_.employeeID is not null then 0 end as clazz\_0\_  from employee employee0\_ left  outer join Skiller employee0\_1\_  on employee0\_.employeeID=employee0\_1\_.employee\_id  left outer join Sales employee0\_2\_  on employee0\_.employeeID=employee0\_2\_.employee\_id  where employee0\_.employeeID=?  所以当使用joined-subclass继承映射策略，当程序查询子类实例时，需要跨越多个字表查询，其复杂度取决于该子类有多少层父类。这样势必会对查询性能有些影响。 |

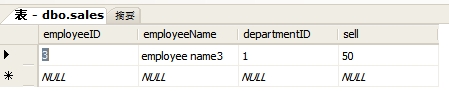
# 采用union-subclass元素的继承映射

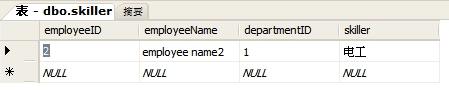
采用这样继承映射策略，父类实例的数据保存在父表中，子类实例的数据仅仅只保存在字表中，没有在附表中有任何记录。在这种继承映射策略下，子表的字段会比父表的字段多。 同时在这种映射策略，既不需要使用辨别者，也不需要使用<key.../>元素来映射共有主键。

|  |
| --- |
| <hibernate-mapping package="com.hibernate.domain">  <class name="Employee" table="employee">  <id name="id" column="employeeID">  <generator class="hilo" />  </id>    <property name="name" column="employeeName" />  <!-- 用于映射N-1关联实体，指定关联实体类为 :Department,指定外键为：departmentID-->  <many-to-one name="department" column="departmentID" />    <!-- 使用union-subclass元素映射Employee类的子类：Skiller -->  <union-subclass name="Skiller" table="skiller">  <property name="skiller" />  </union-subclass>    <!-- 使用union-subclass元素映射Employee类的子类：Sales -->  <union-subclass name="Sales" table="sales">  <property name="sell" />  </union-subclass>  </class>  </hibernate-mapping> |

注：在这种映射策略下，映射持久化类是不能使用identity主键生成策略。

通过添加操作得到如下的两个子类表：





采用这中映射策略，底层数据库的数据看起来更加符合正常情况下的数据库设计，不同实体保存在不同的数据表中。