**Hibernate的主键生成策略**

1. increment: hibernate管理，自动让主键增长，而数据库中主键就可以不用再 设置AUTO\_CREMENT.

**2. identity ： 底层数据库管理，自动让主键增长 MySQL，sql sqlserver**

**3. sequence： 底层数据库管理。 oracle**

4. native hibernate 不管理，让数据库底层自己选择主键如何生成。

5. uuid

6. assigned

**DDL 策略**

**create 启动的时候，先drop，后create**

**create-drop**

**update 启动时，检测表和类是否一致，不一致就会更新。 更新谁呢？ 给老子装犊子**

**validate**

**Hibernate中的三种status**

瞬时status： new

持久status： 在session缓存中

托管status： 从session缓存中移除

**学生和班级**

**<many-to-one name="classes" column="cid"/> </many-to-one>**

**<set name="set">**

**<key column="cid"></key>**

**<one-to-many class="User"/>**

**</set>**

**学生， 课程， 中间表**

**<set name="courseSet" table="student\_course"/>**

**<key column="s\_id"></key>**

**<many-to-many class="domain.Course" column="c\_id"></many-to-many>**

**</set>**

**<set name="studentSet" table="student\_course">**

**<key column="c\_id"></key>**

**<many-to-many class="domain.Student" column="s\_id"></many-to-many>**

**</set>**

**人，身份证**

**<one-to-one name="idCard" constrained="true"/></one-to-one>**

**constrained 我们的主键当外键使用了**

**<one-to-one name="person"></one-to-one>**

<!--和多对一一样，只是增加了一个unique属性，这样就指定了这段唯一>

**<many-to-one name="idCard" column="cardId" unique="true"></many-to-many>**

**<one-to-one name="person" property-ref="idCard" class="domain1.Person"></one-to-one>**

**cascade**

**级联 该实体类对象如果在save或update时，会将这个相关联的对象进行操作**

**<many-to-one name="dept" column="deptId" class="oneToMany.Dept" cacsade="save-update">**

**all. none. save-update. delete. all-delete-orphan**

**inverse 倒转，默认为true， 一端设置**

**维护外键关系**

**维护级联关系**

**Hibernate的三种查询方式**

**1.OID 检索方式**

**session.get()/ session.load()**

**2. HQL 检索方式**

**String sql = "from Staff";**

**Query hqlQuery = session.createQuery(hql);**

**List<Staff> list = hqlQuery.list();**

**for(Staff staff: list){**

System.out.prinltn(staff.toString());

**}**

**3. 本地sql检索方式**

**SQLQuery SqlQuery= session.createSQLQuery("select \* from staff");**

**List list = SqlQuery.list();**

**System.out.println(list);**

**类级别检索策略**

**load和 get的区别**

**1. get是立即加载，load是延迟加载**

**2. get和load都是先从缓存中查找对象，如果有该对象的缓存，则不去数据库中查询，并且返回的是缓存中的状态（是代理对象就返回代理对象，是pojo就返回pojo对象）**

**3. 在缓存中没有对象时，get返回的是pojo对象，load返回的是代理对象**

**4. 在staff.hbm.xml中设置了lazy="false" , 可以让load立即加载**

**关联级别检索策略**

在**<set>标签上或者在<many-to-one>上设置两个属性的值来控制其检索策略， fetch， lazy**

**fetch=join 发送左外迫切连接**

**fetch=select 发送简单的sql语句**

**fetch=subsubselect 生成子查询**

**N+1的问题**

**有5个部门，查询每个部门中的员工，就会发送6条sql语句**

**batch-size 只会发送2条**

**丢失更新， 锁**

**悲观锁，**

**乐观锁。**