1、assigned

主键由外部程序负责生成，在 save() 之前必须指定一个。Hibernate不负责维护主键生成。与Hibernate和底层数据库都无关，可以跨数据库。在存储对象前，必须要使用主键的setter方法给主键赋值，至于这个值怎么生成，完全由自己决定，这种方法应该尽量避免。

<id name="id" column="id">

<generator class="assigned" />

</id>

“ud”是自定义的策略名，人为起的名字，后面均用“ud”表示。

特点：可以跨数据库，人为控制主键生成，应尽量避免。

2、increment

由Hibernate从数据库中取出主键的最大值（每个session只取1次），以该值为基础，每次增量为1，在内存中生成主键，不依赖于底层的数据库，因此可以跨数据库。

<id name="id" column="id">

<generator class="increment" />

</id>

Hibernate调用org.hibernate.id.IncrementGenerator类里面的generate()方法，使用select max(idColumnName) from tableName语句获取主键最大值。该方法被声明成了synchronized，所以在一个独立的Java虚拟机内部是没有问题的，然而，在多个JVM同时并发访问数据库select max时就可能取出相同的值，再insert就会发生Dumplicate entry的错误。所以只能有一个Hibernate应用进程访问数据库，否则就可能产生主键冲突，所以不适合多进程并发更新数据库，适合单一进程访问数据库，不能用于群集环境。

官方文档：只有在没有其他进程往同一张表中插入数据时才能使用，在集群下不要使用。

特点：跨数据库，不适合多进程并发更新数据库，适合单一进程访问数据库，不能用于群集环境。

3、hilo

hilo（高低位方式high low）是hibernate中最常用的一种生成方式，需要一张额外的表保存hi的值。保存hi值的表至少有一条记录（只与第一条记录有关），否则会出现错误。可以跨数据库。

<id name="id" column="id">

<generator class="hilo">

<param name="table">hibernate_hilo</param>

<param name="column">next_hi</param>

<param name="max_lo">100</param>

</generator>

</id>

<param name="table">hibernate_hilo</param> 指定保存**hi**值的表名

<param name="column">next_hi</param> 指定保存**hi**值的列名

<param name="max_lo">100</param> 指定低位的最大值

也可以省略table和column配置，其默认的表为hibernate_unique_key，列为next_hi

<id name="id" column="id">

<generator class="hilo">

<param name="max_lo">100</param>

</generator>

</id>

hilo生成器生成主键的过程（以hibernate_unique_key表，next_hi列为例）：

1. 获得hi值：读取并记录数据库的hibernate_unique_key表中next_hi字段的值，数据库中此字段值加1保存。

2. 获得lo值：从0到max_lo循环取值，差值为1，当值为max_lo值时，重新获取hi值，然后lo值继续从0到max_lo循环。

3. 根据公式 hi * (max_lo + 1) + lo计算生成主键值。

注意：当hi值是0的时候，那么第一个值不是0*(max_lo+1)+0=0，而是lo跳过0从1开始，直接是1、2、3……

那max_lo配置多大合适呢？

这要根据具体情况而定，如果系统一般不重启，而且需要用此表建立大量的主键，可以吧max_lo配置大一点，这样可以减少读取数据表的次数，提高效率；反之，如果服务器经常重启，可以吧max_lo配置小一点，可以避免每次重启主键之间的间隔太大，造成主键值主键不连贯。

特点：跨数据库，**hilo**算法生成的标志只能在一个数据库中保证唯一。

4、seqhilo

与hilo类似，通过hi/lo算法实现的主键生成机制，只是将hilo中的数据表换成了序列sequence，需要数据库中先创建sequence，适用于支持sequence的数据库，如Oracle。

<id name="id" column="id">

<generator class="seqhilo">

<param name="sequence">hibernate_seq</param>

<param name="max_lo">100</param>

</generator>

</id>

特点：与**hilo**类似，只能在支持序列的数据库中使用。

5、sequence

采用数据库提供的sequence机制生成主键，需要数据库支持sequence。如oralce、DB、SAP DB、PostgerSQL、McKoi中的sequence。MySQL这种不支持sequence的数据库则不行（可以使用identity）。

<generator class="sequence">

<param name="sequence">hibernate_id</param>

</generator>

<param name="sequence">hibernate_id</param> 指定**sequence**的名称

Hibernate生成主键时，查找sequence并赋给主键值，主键值由数据库生成，Hibernate不负责维护，使用时必须先创建一个sequence，如果不指定sequence名称，则使用Hibernate默认的sequence，名称为hibernate_sequence，前提要在数据库中创建该sequence。

特点：只能在支持序列的数据库中使用，如**Oracle**。

6、identity

identity由底层数据库生成标识符。identity是由数据库自己生成的，但这个主键必须设置为自增长，使用identity的前提条件是底层数据库支持自动增长字段类型，如DB2、SQL Server、MySQL、Sybase和HypersonicSQL等，Oracle这类没有自增字段的则不支持。

<id name="id" column="id">

<generator class="identity" />

</id>

例：如果使用MySQL数据库，则主键字段必须设置成auto_increment。

id int(11) primary key auto_increment

特点：只能用在支持自动增长的字段数据库中使用，如**MySQL**。

7、native

native由hibernate根据使用的数据库自行判断采用identity、hilo、sequence其中一种作为主键生成方式，灵活性很强。如果能支持identity则使用identity，如果支持sequence则使用sequence。

<id name="id" column="id">

<generator class="native" />

</id>

例如MySQL使用identity，Oracle使用sequence

注意：如果Hibernate自动选择sequence或者hilo，则所有的表的主键都会从Hibernate默认的sequence或hilo表中取。并且，有的数据库对于默认情况主键生成测试的支持，效率并不是很高。

使用sequence或hilo时，可以加入参数，指定sequence名称或hi值表名称等，如

<param name="sequence">hibernate_id</param>

特点：根据数据库自动选择，项目中如果用到多个数据库时，可以使用这种方式，使用时需要设置表的自增字段或建立序列，建立表等。

8、uuid

UUID：Universally Unique Identifier，是指在一台机器上生成的数字，它保证对在同一时空中的所有机器都是唯一的。按照开放软件基金会(OSF)制定的标准计算，用到了以太网卡地址、纳秒级时间、芯片ID码和许多可能的数字，标准的UUID格式为：

xxxxxxxx-xxxx-xxxx-xxxxxx-xxxxxxxxxx (8-4-4-4-12)

其中每个 **x** 是 0-9 或 a-f 范围内的一个十六进制的数字。

<id name="id" column="id">

<generator class="uuid" />

</id>

Hibernate在保存对象时，生成一个UUID字符串作为主键，保证了唯一性，但其并无任何业务逻辑意义，只能作为主键，唯一缺点长度较大，32位（Hibernate将UUID中间的"-"删除了）的字符串，占用存储空间大，但是有两个很重要的优点，Hibernate在维护主键时，不用去数据库查询，从而提高效率，而且它是跨数据库的，以后切换数据库极其方便。

特点：**uuid**长度大，占用空间大，跨数据库，不用访问数据库就生成主键值，所以效率高且能保证唯一性，移植非常方便，推荐使用。

9、guid

GUID：Globally Unique Identifier全球唯一标识符，也称作 UUID，是一个128位长的数字，用16进制表示。算法的核心思想是结合机器的网卡、当地时间、一个随即数来生成GUID。从理论上讲，如果一台机器每秒产生10000000个GUID，则可以保证（概率意义上）3240年不重复。

<id name="id" column="id">

<generator class="guid" />

</id>

Hibernate在维护主键时，先查询数据库，获得一个uuid字符串，该字符串就是主键值，该值唯一，缺点长度较大，支持数据库有限，优点同uuid，跨数据库，但是仍然需要访问数据库。

注意：长度因数据库不同而不同

MySQL中使用select uuid()语句获得的为36位（包含标准格式的"-"）

Oracle中，使用select rawtohex(sys_guid()) from dual语句获得的为32位（不包含"-"）

特点：需要数据库支持查询**uuid**，生成时需要查询数据库，效率没有**uuid**高，推荐使用**uuid**。

10、foreign

使用另外一个相关联的对象的主键作为该对象主键。主要用于一对一关系中。

<id name="id" column="id">

<generator class="foreign">

<param name="property">user</param>

</generator>

</id>

<one-to-one name="user" class="domain.User" constrained="true" />

该例使用domain.User的主键作为本类映射的主键。

特点：很少使用，大多用在一对一关系中。

11、select

使用触发器生成主键，主要用于早期的数据库主键生成机制，能用到的地方非常少。

12、其他注释方式配置

注释方式与配置文件底层实现方式相同，只是配置的方式换成了注释方式

自动增长，适用于支持自增字段的数据库

@Id

@GeneratedValue(strategy = GenerationType.*IDENTITY*)

根据底层数据库自动选择方式，需要底层数据库的设置

如MySQL，会使用自增字段，需要将主键设置成auto_increment。

@Id

@GeneratedValue(strategy = GenerationType.*AUTO*)

使用表存储生成的主键，可以跨数据库。

每次需要主键值时，查询名为**"hibernate_table"**的表，查找主键列**"gen_pk"**值为**"2"**记录，得到这条记录的**"gen_val"**值，根据这个值，和**allocationSize**的值生成主键值。

@Id

@GeneratedValue(strategy = GenerationType.*TABLE*, generator = "ud")

@TableGenerator(name = "ud",

table = "hibernate_table",

pkColumnName = "gen_pk",

pkColumnValue = "2",

valueColumnName = "gen_val",

initialValue = 2,

allocationSize = 5)

使用序列存储主键值

@Id

@GeneratedValue(strategy = GenerationType.*SEQUENCE*, generator = "ud")

@SequenceGenerator(name = "ud",

sequenceName = "hibernate_seq",

allocationSize = 1,

initialValue = 2)

13、小结

1、为了保证对象标识符的唯一性与不可变性，应该让**Hibernate**来为主键赋值，而不是程序。

2、正常使用**Hibernate**维护主键，最好将主键的**setter**方法设置成**private**，从而避免人为或程序修改主键，而使用**assigned**方式，就不能用**private**，否则无法给主键赋值。

2、**Hibernate**中唯一一种最简单通用的主键生成器就是**uuid**。虽然是个**32**位难读的长字符串，但是它没有跨数据库的问题，将来切换数据库极其简单方便，推荐使用！

3、自动增长字段类型与序列

数据库	自动增长字段	序列
MySQL	是	
Oracle		是
DB2	是	是
MS SQL Server	是	
Sybase	是	
HypersonicSQL	是	
PostgreSQL		是
SAP DB		是
HSQLDB	是	
Infomix	是	

4、关于**hilo**机制注意：

hilo算法生成的标志只能在一个数据库中保证唯一。

当用户为**Hibernate**自行提供连接，或者**Hibernate**通过**JTA**，从应用服务器的数据源获取数据库连接时，无法使用**hilo**，因为这不能保证**hilo**单独在新的数据库连接的事务中访问**hi**值表，这种情况，如果数据库支持序列，可以使用**seqhilo**。

5、使用**identity**、**native**、**GenerationType.AUTO**等方式生成主键时，只要用到自增字段，数据库表的字段必须设置成自动增加的，否则出错。

6、还有一些方法未列出来，例如**uuid.hex**，**sequence-identity**等，这些方法不是很常用，且已被其他方法代替，如**uuid.hex**，官方文档里建议不使用，而直接使用**uuid**方法。

7、**Hibernate**的各版本主键生成策略配置有略微差别，但实现基本相同。如，有的版本默认**sequence**不指定序列名，则使用名为**hibernate_sequence**的序列，有的版本则必须指定序列名。

8、还可以自定义主键生成策略，这里暂时不讨论，只讨论官方自带生成策略。

你们都是有经验的开发人员

来源： http://www.cnblogs.com/hoobey/p/5508992.html