Generators:

1. Hibernate supports different types of algorithms for generating primary keys. For each algorithm, there is a separate class.
2. To let hibernate, generate the primary keys then we need to use <generator> class inside mapping file.
3. If we are not mentioning this tag then hibernate by default will assume Assigned as a generator.
4. In case of Assigned generator, programmer is responsible to assign the id value.
5. Except Assigned generator for all other generators, hibernate is responsible to generate primary key values.
6. Hibernate provides different primary key generator classes and all these classes are implemented **org.hibernate.id.IdentifierGenerator**
7. If we want to pass any values to generator then we need to use <param> tag.
8. **Syntax :**
9. <generator class="generator-class-name">
10. <param name="parameter-name"> value </param>
11. </generator>

**assigned :**

* assigned is default generator.
* We if didn't mention any generator then hibernate takes assigned as a generator.
* In this case, programmer is responsible to give primary key value.
* This generator supports in all databases.

**sequence :**

* sequence is database dependent. MySql doesn't support sequence.
* Hibernate is responsible to generate id value. Even if programmer passes the id value, but hibernate won't take the programmer given value.
* Hibernate will execute the sequence and assigns the value to the object when inserting a new record.
* We need to pass our sequence to <generator> class.
* Syntax :
* <id name="employeeId" column="EID">
* <generator class="sequence">
* <param name="sequence">SequenceName</param>
* </generator>
* </id>
* If we didn't pass any sequence then hibernate will take default sequence. The default sequence taken by hibernate is hibernate\_sequence.

**increment :**

* Increment generator is database independent.
* Increment generator will generate id value based on formula **Max of id value + 1**
* It there are no records in database, then increment will generate 1 as an id value.

**Note :**POJO class, Mapping and Configuration files are same as above application.

**identity :**

* Identity is database dependent. Oracle database doesn't support identity generator.
* Identity generator is similar to increment generator, but the difference is, increment generator is database independent and hibernate uses a select operation for selecting max of id before inserting new record.
* But in case of identity, no select operation will be performed to insert id value for new record by hibernate.
* Here id value is generated by the database but not by hibernate.

Syntax :

<id name="employeeId" column="EID">

<generator class="identity" />

</id>

**Note :**POJO class, Mapping and Configuration files are same as above application.

Oracle doesn't support identity. So we will get exception if we use oracle database.

**hilo :**

* hilo is database independent.
* hilo will insert 1 as a id value for first record. For the second record it will insert 32768 as id value. And from next record it will increment the old id value by 32768 and stores the new value.
* To store the count of id values, hibernate will use a separate table known as hibernate\_unique\_value. default column name in this table is next\_hi.
* To create our own table and column, we need to pass 2 parameters for hilo generator.

Syntax :

<id name="employeeId" column="EID">

<generator class="hilo">

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

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

</generator>

</id>

**Note :**POJO class, Mapping and Configuration files are same as above application.

**native :**

native doesn't have a separate generator class. It just uses other generator classes. **identity, sequence, hilo**

* First it will check whether the database supports identity or not.
* If database not support identity then it will go for sequence.
* If database not support sequence then it will go for hilo.

**Note :**POJO class, Mapping and Configuration files are same as above application.

**foreign :**

foreign generator is used to one-to-one relationship.