Primary Key Generation Strategies

Consider a requirement where an admin should be able to add the customer details to the database and customer id should be automatically generated.

To implement this requirement, some mechanism is required so that the primary key gets generated automatically. JPA provides different strategies using which primary key values can be generated automatically. The different strategies provided by JPA are as follows:

1. IDENTITY strategy
2. TABLE strategy
3. SEQUENCE strategy
4. AUTO strategy

Note: These strategies are not supported by all databases. For example, IDENTITY strategy is not supported by Oracle database because Oracle does not provide identity columns. Similarly, SEQUENCE is not supported by MySQL database because MySQL does not provide support for database sequences.

Strategy	MySQL	Oracle SQL
Identity	✓	--
Table	✓	✓
Sequence	--	✓
Auto	✓	✓

Identity Strategy

In this strategy, the value of the primary key is generated using the identity column of the table. The identity column of a table is an integer column having a primary key or unique constraint with the AUTO_INCREMENT feature. For example, in following customer table customer_id is an identity column:

```
1. CREATE TABLE customer (
2.   customer_id int AUTO_INCREMENT,
3.   email_id varchar(20),
4.   name varchar(10),
5.   date_of_birth date,
6.   constraint ps_customer_id_pk primary key (customer_id)

7. );
```

In this strategy, if there are no rows in the table, the starting value of the identity column will be 1, and it will be incremented by 1 for each new row added. If the rows are already present in the table, then the maximum value from the identity column is taken and incremented by 1 to generate the next primary key value.

To use this strategy, the primary key attribute of the entity class is annotated with @GeneratedValue annotation with GenerationType.IDENTITY as the value of the strategy attribute along with @Id annotation. The following code snippet shows how @GeneratedValue annotation is used to generate values for customerId:

```
1.  @Entity
2.  public class Customer{
3.    @Id
4.    @GeneratedValue(strategy=GenerationType.IDENTITY)
5.    private Integer customerId;
6.
7.       // rest of the code


8.  }
```

When you persist a new Customer entity, JPA selects customer_id column of customer table to generate the primary key value. If no rows are present in customer table then the generated value of customerId will be **1**. If rows are already present is customer table then maximum value present in customer_id column will be used for generating value of customerId. For example, if maximum value of customer_id column is 5, then the generated primary key value will be 6.

Table Strategy

In this strategy, following **hibernate_sequences** database table is used to generate primary key values:

sequence_name	next_val
default	primary key value

In this table the value in sequence_name column is default and the next_val column contains the value of the primary key. If this table is not present in database you must create it.

To use this strategy, the primary key attribute of entity class is annotated with @GeneratedValue annotation with GenerationType.TABLE as the value of the strategy attribute along with @Id annotation. The following code snippet shows how @GeneratedValue annotation is used to generate values for customerId:

```
1.  @Entity
2.  public class Customer{
3.      @Id
4.      @GeneratedValue(strategy=GenerationType.TABLE)
5.      private Integer customerId;
6.      //rest of the code


7.  }
```

When you persist a new Customer entity, JPA generates next primary key value from the value present in next_val column by incrementing it by 1. For example, if the value in the next_val column is 5 then the primary key value generated for customerId will be 6 and the value in the next_val column will be updated to 6.

You have seen how to generate primary key values using table strategy using the default table. But you can also specify your own table for generating primary key values. To do this you have to use @TableGenerator annotation is used along with @GeneratedValue and @Id annotation to define which table has to be used for generating primary key values.

Table Strategy using custom table

Consider the following id_gen table:

gen_key	gen_value
cust_id	primary key value

To use this table for generating primary key values for customerId @TableGenerator annotation is used as follows:

```
1.  @Entity
2.  public class Customer{
3.    @Id
4.    @TableGenerator(
5.              name="pkgen",
6.              table="id_gen",
7.              pkColumnName="gen_key",
8.              valueColumnName="gen_value",
9.              pkColumnValue="cust_id",
10.             allocationSize=1)
11.   @GeneratedValue(generator="pkgen",strategy=GenerationType.TABLE)
12.   private Integer customerId;
13.     //rest of the code


14. }
```

When you persist a new Customer entity, JPA generates next primary key value from the value present in gen_value column by incrementing it by 1. For example, if the value in the gen_value column is 5 and you persist a new Customer entity object then customer details with customerId as 6 will be persisted in the customer table and the value in the gen_value column will be updated to 6.

The @TableGenerator annotation has the following attributes:

- **name :** It specifies the name of the generator. This name is used to access the generator inside the entity class.
- **table :** It specifies the name of the table which has to be used for generating primary key values. In our case the table is id_gen.
- **pkColumnName :** It is the name of the primary key column which contains generator names used for generating primary key value. In our case pkColumnName is gen_key.
- **valueColumnName :** It is the name of the column which contains the last primary key value generated. In our case valueColumnName is gen_value.
- **pkColumnValue :** In table, many generators are may be present. This attribute specifies which generator has to be used for generating primary key value among a set of generators in the table. In our case pkColumnValue is cust_id.
- **allocationSize :** It specifies the amount to increment by when allocating id numbers from the generator.

Sequence and Auto Strategies

Sequence strategy

In the above code, no information is provided about which database sequence to use so default Hibernate sequence, hibernate_sequence, is used. This sequence does not come with the database. The developer has to create it.

You can also use a custom database sequence for generating primary key values. For this, you have to define a sequence generator which contains information about database sequence using @SequenceGenerator annotation. Then a reference of this sequence generator is used in @GeneratedValue annotation. For example, the following code snippet shows how custom database sequence can be used:

```
1.  @Entity
2.  public class Customer{
3.    @Id
4.
    @SequenceGenerator(name="pkgen",sequenceName="customer_sequence",allocationS
    ize=1)
5.    @GeneratedValue(generator="pkgen",strategy=GenerationType.SEQUENCE)
6.    private Integer customerId;
7.    // rest of the code


8.  }
```

In the above code, @SequenceGenerator annotation defines a sequence generator with the name "pkgen" which references database sequence named "customer_sequence" and the @GeneratedValue annotation references this using generator attribute. The customer_sequence sequence must be present in the database. If it is not present you must create it.

Auto strategy

n this strategy, the persistence provider automatically selects an appropriate strategy for generating primary key values depending on the database used. For example, if Hibernate 5.0 is the persistence provider and MySQL database is used, then this strategy selects the TABLE strategy (hibernate_sequences table) for generating primary key values otherwise it uses SEQUENCE strategy (hibernate_sequence sequence). To use this strategy, the primary key attribute of an entity class is annotated with @GeneratedValue annotation with GenerationType.AUTO as the value of the strategy attribute along with @Id annotation as follows:

```
1.  @Entity
2.  public class Customer {
3.     @Id
4.     @GeneratedValue(strategy=GenerationType.AUTO)
5.     private int customerId;
6.     //rest of the code


7.  }
```

Note: If you do not mention strategy then AUTO strategy is used by default.