Generate Identifiers

JPA Specification supports four pk generation strategy

- Options to generate primary keys
    - GenerationType.AUTO - default Type
    - GenerationType.IDENTITY
    - GenerationType.SEQUENCE
    - GenerationType.TABLE

If you use hibernate as persistence provider it selects generation strategy based on database specific dialect.

Most database supports generation type - **SEQUENCE**

For Mysql -
Auto - Create a seperate table named Hibernate_sequence to maintain the identity of the primary key. And gives the idea of next_value coming.
Table 1) demo(id,name)
Table 2) hibernate_sequences(next_val)

Hibernate
```
  create table demo (
       id bigint not null,
        name varchar(255),
        primary key (id)
    ) engine=InnoDB
```
Hibernate:
```
    create table hibernate_sequence (
       next_val bigint
    ) engine=InnoDB
```
Hibernate:
```
insert into hibernate_sequence values ( 1 )
```

Identity - Does not create a new table to maintain an identity of primary key and increment the primary key automatically.
Best Options for MySql

Hibernate disables insert batching at the JDBC level transparently if you use an identity identifier generator.

Batch Processing allows you to group related SQL statements into a batch and submit them with one call to the database. When you send several SQL statements to the database at once, you reduce the amount of communication overhead, thereby improving performance.

Single Table - demo(id,name)

Hibernate:

```
    create table demo (
       id bigint not null auto_increment,
        name varchar(255),
        primary key (id)
    ) engine=InnoDB
```

Sequence - Create a seperate table named Hibernate_sequence to maintain the identity of the primary key. And gives the idea of upcoming next_value .

With this , we can perform JDBC Batching ,also it allows hibernate when to perform the insert operation.

Fastest And Recommendation Option.

Table 1) demo(id,name)

Table 2) hibernate_sequences(next_val)

Hibernate:

```
    create table demo (
       id bigint not null,
        name varchar(255),
        primary key (id)
    ) engine=InnoDB
Hibernate: 

    create table hibernate_sequence (
       next_val bigint
    ) engine=InnoDB
Hibernate: 

    insert into hibernate_sequence values ( 1 )
```

Table - Create a seperate table named Hibernate_sequences to maintain the identity of the primary key. In that separate table you will see two column

next_val=current_pk

Slowest option

Don't use it.

Table 1) demo(id,name)

Table 2) hibernate_sequences(sequence_name,next_val)

```
Hibernate: 
    create table demo (
       id bigint not null,
        name varchar(255),
        primary key (id)
    ) engine=InnoDB
Hibernate: 

    create table hibernate_sequences (
       sequence_name varchar(255) not null,
        next_val bigint,
        primary key (sequence_name)
    ) engine=InnoDB
Hibernate: 

 insert into hibernate_sequences(sequence_name, next_val) values ('default',0)
```

UUID - Universally Unique identifiers

UUIDs are 128-bit hexadecimal numbers that are globally unique.
The problem with this is that they are very big in size and don't index well.

Note : MySQL does not support custom sequences. So if you are using mysql and want to avoid auto-increment ids, your only bet is server side.

- **UUID has performance impact on indexing**
- **Auto-increment keys are not possible in sharded databases**

Note : Oracle doesn't support IDENTITY