**21.1. Primary Keys – Overview**

**Primary key**:

1. Primary key is a uniquely identifying each row in a table.
2. Must be a unique value
3. Cannot contain NULL values

**MySQL - Auto Increment**:

In MySQL database generally when we are creating a table, we make the id as primary key and make it auto increment.

CREATE TABLE student (  
id int(11) NOT NULL AUTO\_INCREMENT,  
first\_name varchar(45) DEFAULT NULL,  
last\_name varchar(45) DEFAULT NULL,  
email varchar(45) DEFAULT NULL,  
PRIMARY KEY (id)  
)

**Hibernate Identity - Primary Key**:

In Hibernate when we build a class we may use of an annotation, **@Id.** This **@Id** basically tells Hibernate, thisgiven field is a primary key. This field maps the column in the database table, and the column name is id. We basically leave it up to the database to actually generate a primary key for us.

@Entity  
@Table(name="student")  
**public class** Student {  
 **@Id  
 @Column(name="id")** private int id;  
 **…**}

**Hibernate Identity - Primary Key**:

If we want to generate password explicitly, we can tell hibernate how to actually perform generation. We have to give a strategy to hibernate for generating that ID. If we don’t specifying anything, by default, hibernate use the appropriate strategy for the given database implementation.

**ID Generation Strategies**:

Some “**ID Generation Strategies**” in hibernate are given bellow

|  |  |
| --- | --- |
| **Name** | **Description** |
| **GenerationType.AUTO** | Pic an appropriate strategy for the particular database |
| **GenerationType.IDENTITY** | Assign primary keys using database identity column |
| **GenerationType.SEQUENCE** | Assign primary keys using a database sequence |
| **GenerationType.TABLE** | Assign primary keys using an underlying database table to ensure uniqueness |

The most common strategy in MySQL is “**GenerationType.IDENTITY** “.

**public** **class** Student {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "id")

**private** **int** id;

.......

}

**CUSTOM generation strategy**:

We can define our own CUSTOM generation strategy. We can define our own ID using our own custome business logic.

1. Create subclass **of org.hibernate.id.SequenceGenerator**
2. Override the method: **public Serializable generate(…)**

**Primary Keys - Changing the Starting Index**:

We can change the auto increment values. We can use a small SQL to modify the MySQL database.

**SQL statement**:

alter table hb\_student\_tracker.student auto\_increment = 3000

Now all the new entries into the database will start at 3000.

**Value in the table**:

'1', 'Md. Ruhul Amin', 'Ruhul', 'ruhul@gmail.com'

'2', 'Md. Rezaul Islam', 'Reza', 'reza@gmail.com'

'3', 'Md. Ariful Islam', 'Arif', 'arif@gmail.com'

'4', 'Md. Rafiul Islam', 'Rafi', 'rafi@gmail.com'

'3000', 'Md. Ruhul Amin', 'Ruhul', 'ruhul@gmail.com'

'3001', 'Md. Rezaul Islam', 'Reza', 'reza@gmail.com'

'3002', 'Md. Ariful Islam', 'Arif', 'arif@gmail.com'

'3003', 'Md. Rafiul Islam', 'Rafi', 'rafi@gmail.com'

**Reset database table**:

Now we reset our table for start primary key from one.

**SQL statement**:

truncate hb\_student\_tracker.student;

This statement deletes all data from the database and reset the sequence to one.

21.1. Primary Keys - Overview