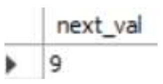
Sequence Table

Sequence Table: Hibernate uses a separate database table to simulate a sequence generator. This table typically has two columns: one for the sequence name and another for the next value in the sequence.

Entity ID Generation: When you save a new entity with an assigned ID, Hibernate checks the sequence table to obtain the next available ID for that entity type.

Optimistic Locking: To ensure that multiple transactions don't get the same ID, Hibernate uses optimistic locking mechanisms. It incrementally updates the sequence table and assigns the new value to the entity being saved.



next_val
9

In Oracle databases, Hibernate typically uses database sequences for ID generation rather than a separate sequence table. Oracle provides a builtin feature called **"sequences"** that can be used to generate unique identifier values.

When we configure Hibernate for an Oracle database, we would typically use the **@SequenceGenerator** annotation to specify the details of the sequence, and then use the **@GeneratedValue** annotation on the ID field of your entity to indicate that the ID should be generated using the specified sequence.

example in Java using JPA annotations and Hibernate with Oracle:

```
@Entity
```

```
@Table(name = "employee")

@SequenceGenerator(name = "emp_seq", sequenceName = "EMPLOYEE_SEQ", allocationSize = 1)

public class Employee {

   @Id

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

   @Column(name = "employee_id")

   private Long id;

}
```

@SequenceGenerator is used to define the details of the sequence, such as its name (your_entity_seq), the sequence name in the database (EMPLOYEE_SEQ), and the allocation size.

@GeneratedValue with GenerationType.SEQUENCE indicates that the ID should be generated using the specified sequence (EMPLOYEE_SEQ).

Hibernate will use the Oracle sequence to generate a unique ID for that entity. There is no need for a separate sequence table in this case.