educación IT
CODE YOUR FUTURE

Bootcamp Java Developer

Fase 3 - Java Architect

Módulo 30

Mapeo de herencia de clases

Problemática

No es posible modelar la herencia en un modelo de datos. Esto produce un *gap* entre el modelo relacional y el modelo de objetos. SQL no provee soporte para herencia.

Soluciones

Una tabla por clase concreta

Esta propuesta es la solución más sencilla. Se utiliza una clase concreta por cada tabla. **Por cada atributo de la clase, existe un campo en la tabla correspondiente**.

Es la solución ideal si no se tienen en cuenta conceptos como herencia y polimorfismo. Descarta el uso de relaciones de herencia y polimorfismo del modelo relacional.

Una tabla por subclase

Representa la herencia a través de claves foráneas, ya que por cada subclase existe una tabla. Cada subclase, incluyendo clases abstractas, tiene su propia tabla. Cada tabla contiene campos solo por cada atributo que no es heredado.

La clave primaria es, a su vez, la clave foránea correspondiente a la superclase. Para obtener los datos de una subclase, es necesario hacer un join entre la tabla que representa a la superclase y la tabla que representa a la subclase.

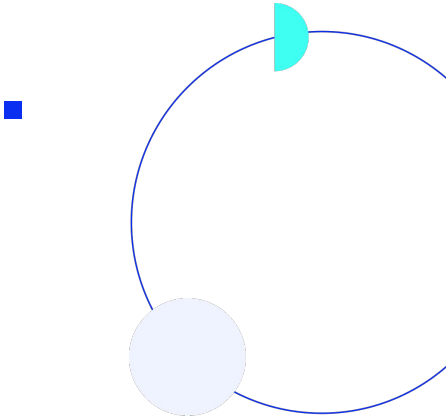
Una tabla por jerarquía de clases

Se construye una tabla por cada jerarquía de clases, donde la tabla posee un campo por cada atributo de la jerarquía de clases.

Adicionalmente, se utiliza un campo de tipo discriminador (*discriminator*) para establecer qué tipo de clase corresponde a cada registro.

Integra el concepto de polimorfismo y es la mejor estrategia en términos de *performance* y simplicidad.

La única restricción que posee es que los atributos de las subclases deben quedar como campos que pueden estar en `NULO`.



Para confeccionar esta relación, dentro del modelo relacional será necesario construir la siguiente tabla denominada transportes:

Column Name	Datatype	NOT NULL
tr_id	INTEGER	✔
tr_ancho	INTEGER	✔
tr_largo	INTEGER	✔
tr_au_patente	VARCHAR(45)	✔
tr_discriminador	VARCHAR(45)	✔

```
[code]
@Entity
@Table(name = "transportes")
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(
    name="tr_discriminador",
    discriminatorType=DiscriminatorType.STRING
)
@DiscriminatorValue(value="T")
public class Transporte {

    @Id
    @GeneratedValue
    @Column(name = "tr_id")
    private Long transporteId;

    @Column(name = "tr_largo")
    private int largo;
```

...

```
@Column(name = "tr_ancho")
    private int ancho;

// Constructors and Getter/Setter methods,
}

-------------------------------------------------
@Entity
@Table(name="transportes")
@DiscriminatorValue("A")
public class Auto extends Transporte {

    @Column(name="tr_au_patente")
    private String patente;
    // Constructors and Getter/Setter methods,
}
[/code]
```

El código presentado en *bold* es el código necesario para representar la **relación de herencia dentro de Hibernate**.

El discriminador se utiliza para guardar un valor en la tabla que identificará si un registro corresponde a un `Transporte` o a un `Auto`.

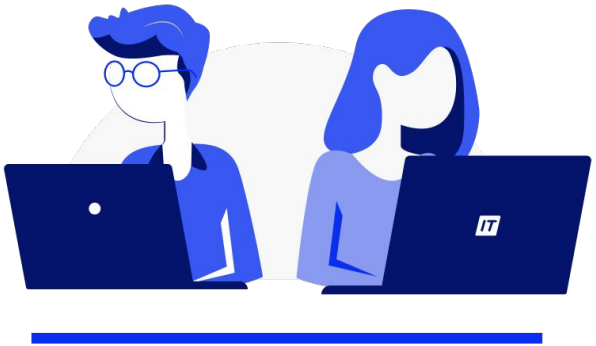
La clase `Transporte` es la superclase, por tal motivo los valores para las *annotations* `@Inheritance` y `@DiscriminatorColumn` se definen en ella.

@Inheritance

Define la estrategia a utilizar, se define en la superclase de la jerarquía. Los valores posibles son `JOINED`, `SINGLE_TABLE` y `TABLE_PER_CLASS`.

@DiscriminatorValue

Se utiliza para especificar el valor de la columna discriminadora para las entidades de cada tipo. La annotation **`DiscriminatorValue`** puede ser solamente especificada en una clase concreta.



@DiscriminatorColumn

Se utiliza para definir la columna que será utilizada como discriminador, se utiliza en las estrategias **`SINGLE_TABLE`** ay **`JOINED`**. La estrategia de mapeo y la columna del discriminador se especifican solamente en la superclase de la jerarquía.

Si la *annotation* para `@DiscriminatorColumn` no está definida, y una columna para el discriminador es necesaria, los valores por defecto son "`DTYPE`" para el nombre y `DiscriminatorType.STRING` para el tipo.

¡Sigamos trabajando!