Actividad JPA Punto 1

Brandon Javier Escudero Avila

Facultad de ingeniería

Fundación Universitaria Compensar

DESARROLLO DE SOFTWARE WEB BACK-END

Carlos Hugo Neiva Reyes

Marzo 8 de 2025

DOCUMENTO DE LECTURA: INTRODUCCIÓN A JPA Y RELACIONES ENTRE ENTIDADES

¿Qué es JPA?

La **Java Persistence API (JPA)** es una especificación de Java que define cómo interactuar con bases de datos relacionales desde aplicaciones Java. JPA no es un framework en sí mismo, sino una serie de reglas y anotaciones que frameworks como Hibernate, EclipseLink o OpenJPA implementan.

Con JPA, los desarrolladores pueden mapear clases Java a tablas en una base de datos, permitiendo que los objetos se guarden, consulten, actualicen o eliminen fácilmente.

¿Qué es una Entidad?

En JPA, una **entidad** es una clase Java que representa una tabla de base de datos. Cada instancia de la clase representa una fila de esa tabla. Para que una clase sea considerada entidad, debe llevar la anotación:

```
@Entity
public class Libro {
   @Id     private Long id;     private String titulo;
}
```

Tipos de Relaciones en JPA

Las relaciones permiten conectar entidades entre sí, simulando las asociaciones entre tablas en la base de datos. JPA define **cuatro tipos principales de relaciones**:

1. OneToOne (Uno a Uno)

Ejemplo: Un proyecto tiene un único evaluador asignado.

```
@OneToOne
@JoinColumn(name = "evaluador_id") private
Evaluador evaluador;
```

2. OneToMany (Uno a Muchos)

Ejemplo: Un autor puede tener varios libros publicados.

```
@OneToMany(mappedBy = "autor")
private List<Libro> libros;
```

3. ManyToOne (Muchos a Uno)

Ejemplo: Muchos libros pueden pertenecer a una misma categoría.

```
@ManyToOne
@JoinColumn(name = "categoria_id") private
Categoria categoria;
```

4. ManyToMany (Muchos a Muchos)

Ejemplo: Un libro puede pertenecer a varias categorías y una categoría puede tener varios libros.

```
        @ManyToMany
        @JoinTable(
            name = "libro_categoria",     joinColumns =
@JoinColumn(name = "libro_id"),     inverseJoinColumns =
@JoinColumn(name = "categoria_id")
        )
        private List<Categoria> categorias;
```

Anotaciones clave de JPA

Anotación	**Función**
@Entity	Define una clase como entidad.
@Table	Define el nombre de la tabla.
@Id	Define la clave primaria.
Anotación	**Función**
@GeneratedValue	Configura la estrategia de generación de la clave primaria.
@Column	Configura propiedades de una columna específica.
@OneToOne	Define una relación uno a uno.
@OneToMany	Define una relación uno a muchos.
@ManyToOne	Define una relación muchos a uno.
@ManyToMany	Define una relación muchos a muchos.
@JoinColumn	Define la columna que actúa como clave foránea.

@JoinTable	Define una tabla intermedia para relaciones muchos a muchos.

Importancia de las Relaciones

En aplicaciones reales, las entidades no existen aisladas. Los sistemas de información funcionan con datos relacionados, por ejemplo:

- Un cliente realiza varios pedidos.
- Un pedido incluye varios productos.
- Cada producto pertenece a una categoría.

Estas relaciones deben mapearse correctamente para que la base de datos refleje correctamente el modelo de negocio.

Validación de Relaciones

Al definir relaciones, es importante: Configurar correctamente las **claves foráneas** (JoinColumn).

- Definir el lado dueño y el lado inverso (mappedBy).
- Manejar correctamente el **cascading** para propagación de operaciones (persist, merge, remove).
- Configurar la **carga (fetch)**: Lazy o Eager, según el contexto.

Ejemplo completo: Entidad Libro y su relación con Autor y Categoría

```
@Entity
public class Libro {
```

```
        @Id
        @GeneratedValue(strategy =
GenerationType.IDENTITY)
        private Long id;
```

```
        private String titulo;
```

```
        @ManyToOne
        @JoinColumn(name = "autor_id")
        private Autor autor;
```

```
        @ManyToMany
        @JoinTable(     name = "libro_categoria",
joinColumns = @JoinColumn(name = "libro_id"),
inverseJoinColumns = @JoinColumn(name = "categoria_id")
        )
        private List<Categoria> categorias;
    }
```

Glosario sugerido (completa tú mismo tras leer)

Término	**Definición (según lo entendido)**
Entidad	En una clase en java que representa una tabla de bases de datos.
Relación	Es una conexión

	que permite conectar entidades entre ellas.
Término	**Definición (según lo entendido)**
Clave primaria	Es un campo de la entidad que identifica de forma única a las filas.
Clave foránea	Conjunto de columnas en una entidad que hace referencia a una clave primaria de otra entidad que las relaciona
OneToOne	Es una relación entre entidades con un único registro asignado.
OneToMany	Es una relación entre entidades donde una entidad tiene varios registros asignados.

JoinColumn	Anotación que indica qué columna almacena la relación con otra tabla.
Cascading	Permite que los cambios en una entidad también afecten a sus relaciones
Fetch	ndica si los datos relacionados se cargan de inmediato o solo cuando se necesitan
Persistencia	Guardar datos en la base de datos para que no se pierdan cuando se apaga la aplicación

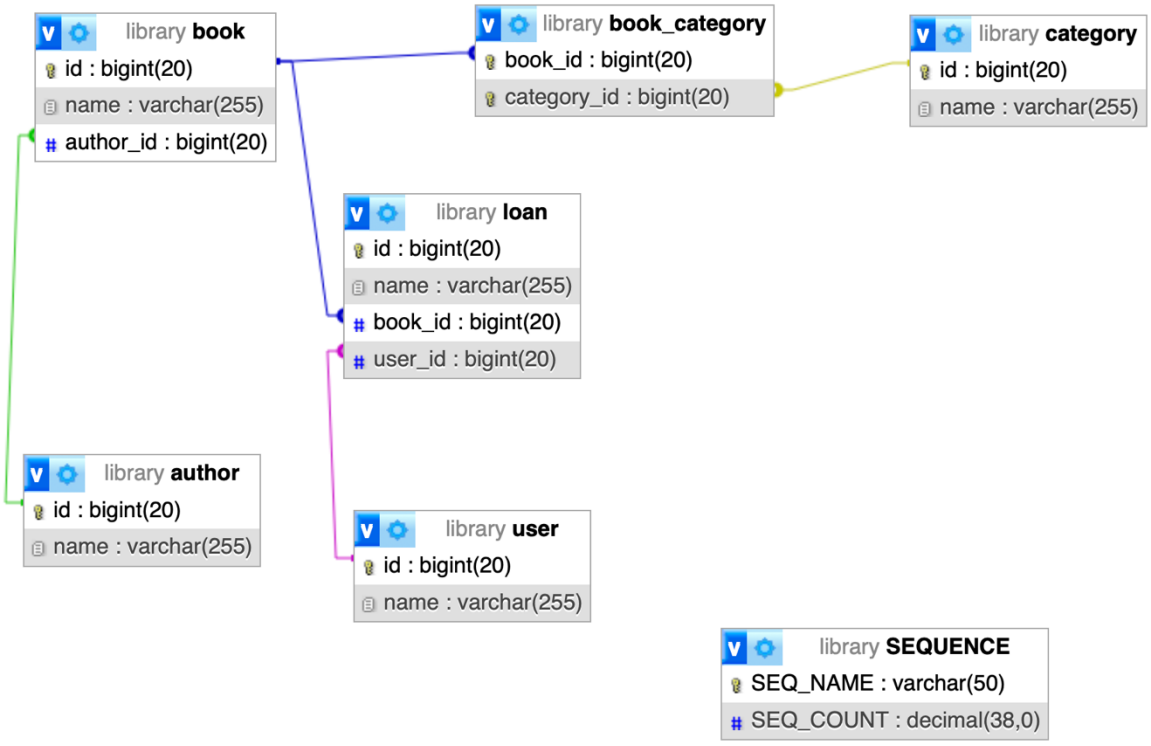
Glosario de términos

1. **Entidad**: Representa una tabla en la base de datos.
2. **Clave primaria (@Id)**: Identificador único de cada registro en la base de datos.
3. **Relación @ManyToOne**: Varios registros pueden estar relacionados con un solo registro en otra tabla.
4. **Relación @OneToMany**: Un registro puede tener varios registros asociados en otra tabla.
5. **Relación @ManyToMany**: Se usa cuando varios registros de una tabla pueden estar relacionados con varios registros de otra.
6. **@JoinColumn**: Define la columna que se usará para la clave foránea.
7. **@JoinTable**: Indica una tabla intermedia en relaciones @ManyToMany.

Explicación de las relaciones

- **Libro - Autor (@ManyToOne)**: Un libro solo puede tener un autor, pero un autor puede haber escrito varios libros.
- **Libro - Ejemplares (@OneToMany)**: Un libro puede tener varias copias físicas en la biblioteca.
- **Libro - Categorías (@ManyToMany)**: Un libro puede pertenecer a varias categorías, y una categoría puede contener varios libros.
- **Usuario - Préstamos (@OneToMany)**: Un usuario puede realizar varios préstamos.
- **Préstamo - Ejemplar (@ManyToOne)**: Cada préstamo está asociado a un único ejemplar de un libro.
- **Préstamo - Usuario (@ManyToOne)**: Cada préstamo está ligado a un solo usuario.

Diagrama de biblioteca creado por base de datos en la siguiente pagina.


library book
id : bigint(20)
name : varchar(255)
author_id : bigint(20)
library book_category
book_id : bigint(20)
category_id : bigint(20)
library category
id : bigint(20)
name : varchar(255)
library loan
id : bigint(20)
name : varchar(255)
book_id : bigint(20)
user_id : bigint(20)
library author
id : bigint(20)
name : varchar(255)
library user
id : bigint(20)
name : varchar(255)
library SEQUENCE
SEQ_NAME : varchar(50)
SEQ_COUNT : decimal(38,0)