

Escuela Java – JPA

Parte 1



TOGETHER. FREE YOUR ENERGIES

Overview I



Un object model usa principios de abstracción, encapsulamiento, modularidad, jerarquía, tipado, concurrencia y polimorfismo para construir aplicaciones.

Vs.



Un modelo relacional define la estructura de los datos, la forma de manipularlos y la integridad de los mismos.



Overview II

Ah, pero yo tenía JDBC para comunicarme desde Java con la BD!

Overview III

Las conexiones. Tengo que manejar las tóxicas conexiones a pata, que me disminuyen la esperanza de vida

El horrible código que te queda.

Las asociaciones. Decime, qué hago con las asociaciones?

Y aparte SQL no es estándar ni a martillazos.



ur energies

Overview IV

Entonces...



Together. Free your energies

Overview V

Ventajas de un ORM



Productividad: Como utilizamos metadatos para persistir y consultar datos, el tiempo de desarrollo disminuye y la productividad aumenta.






Mantenibilidad: Como gran parte del trabajo es realizado a través de configuración, escribimos menos código.



Vendor independence: Un ORM se abstrae del DBMS y del dialecto SQL. Esto nos da portabilidad para soportar múltiples motores.

Overview VI

Desventajas de un ORM

-  **Learning curve:** Hay que aprender a realizar los mapeos y posiblemente un nuevo lenguaje de consulta.
-  **Overhead:** Para una aplicación muy sencilla un ORM puede constituir extra overhead.
-  **Performance:** Para grandes batch updates, hay impacto en la performance

Algoritmo

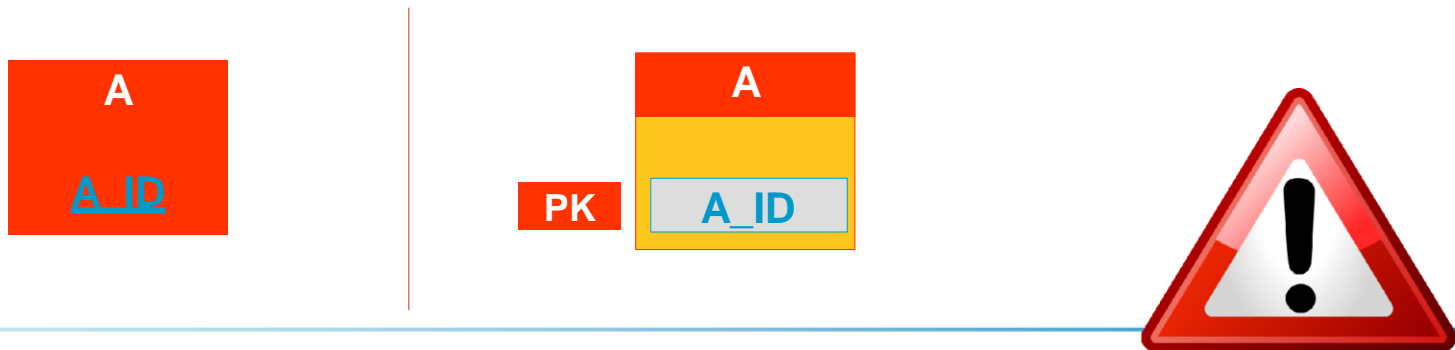
El Algoritmo de mapeo general



Para cada clase, crear una tabla con el mismo nombre.

Poner para cada atributo de la clase un campo en la tabla con el mismo nombre.

Crear una Primary Key para el atributo que 'marcamos' como ID en la clase



Algoritmo I

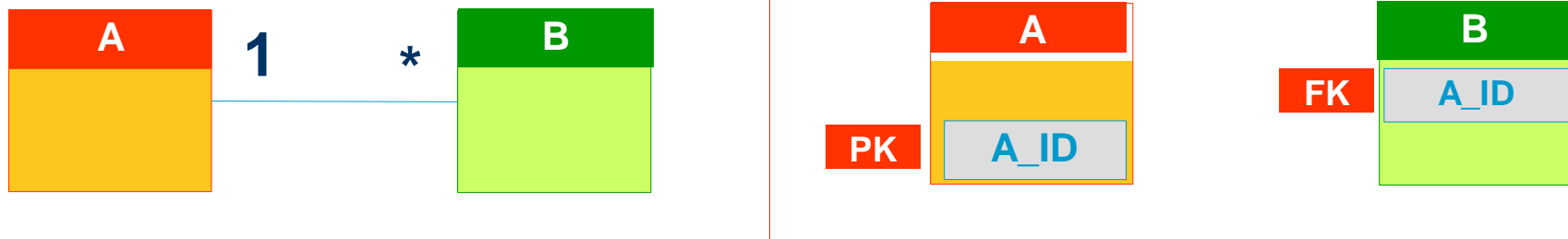


Para cada relación uno a uno

Colocar un campo de foreign key en la tabla A o B según corresponda por navegabilidad

Colocar el campo como NOT NULL si la relación es con exactamente 1 elemento

Algoritmo II

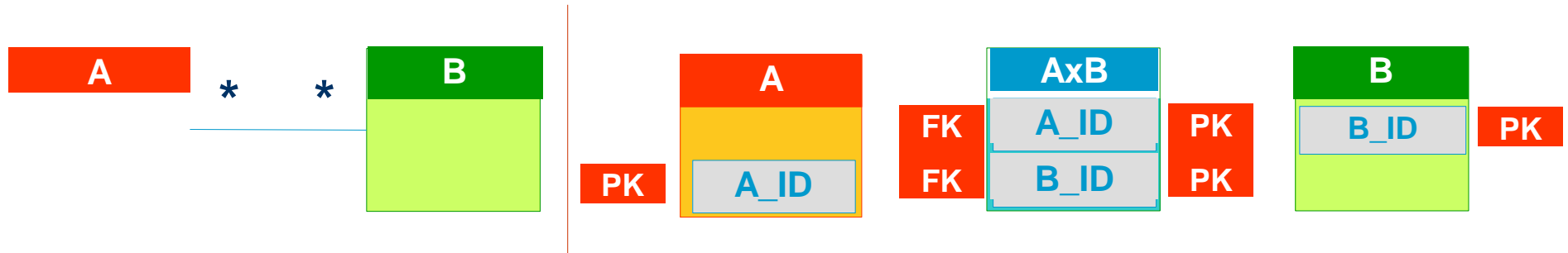


Para cada relación uno a muchos

Colocar un campo de foreign key a la tabla A en la tabla B

Colocar el campo como NOT NULL si la relación es con 1 elemento como mínimo

Algoritmo III

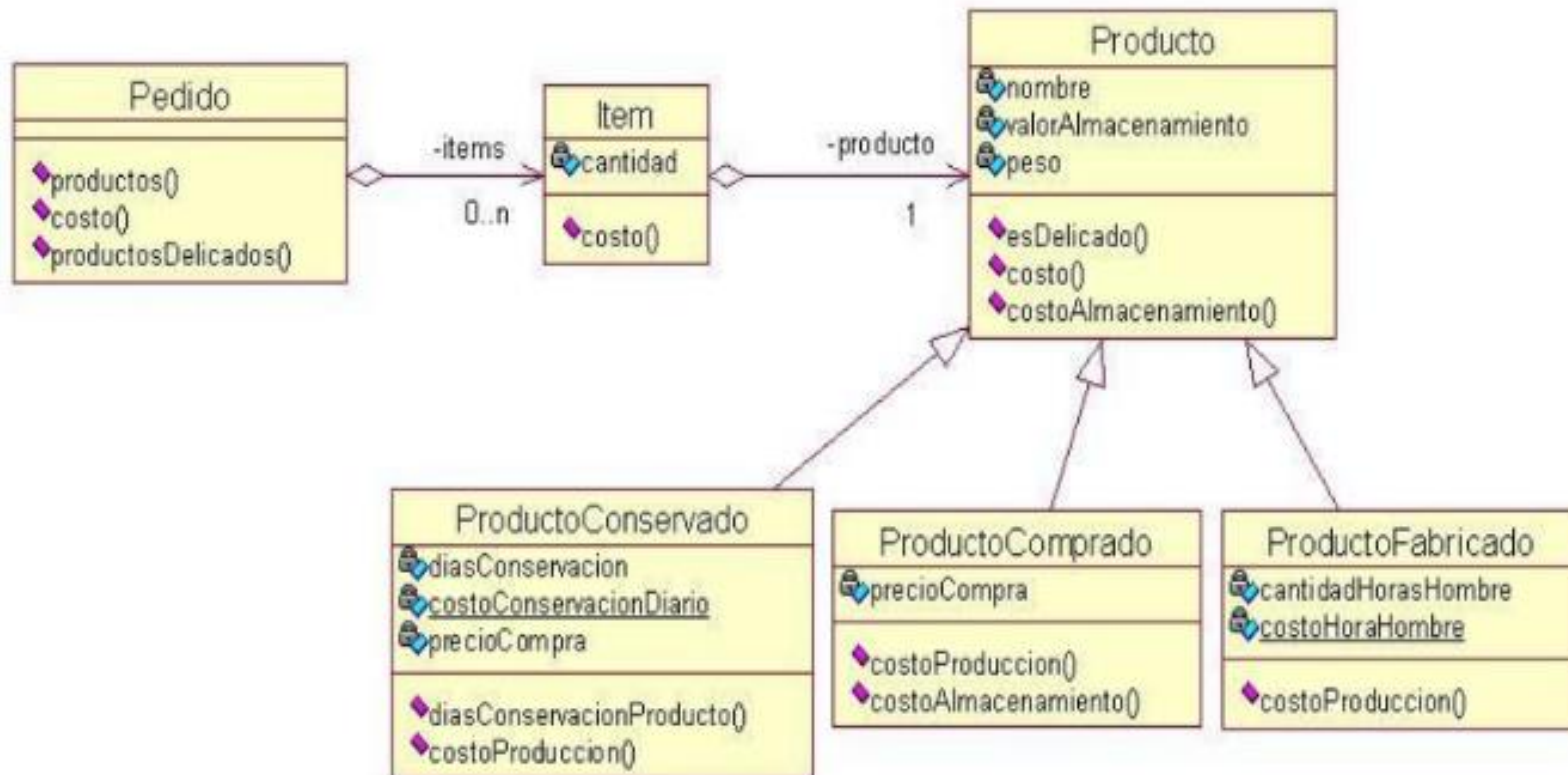


Para cada relación muchos a muchos

Crear una tabla intermedia AxB, con una foreign key a la tabla A y otra a la tabla B.

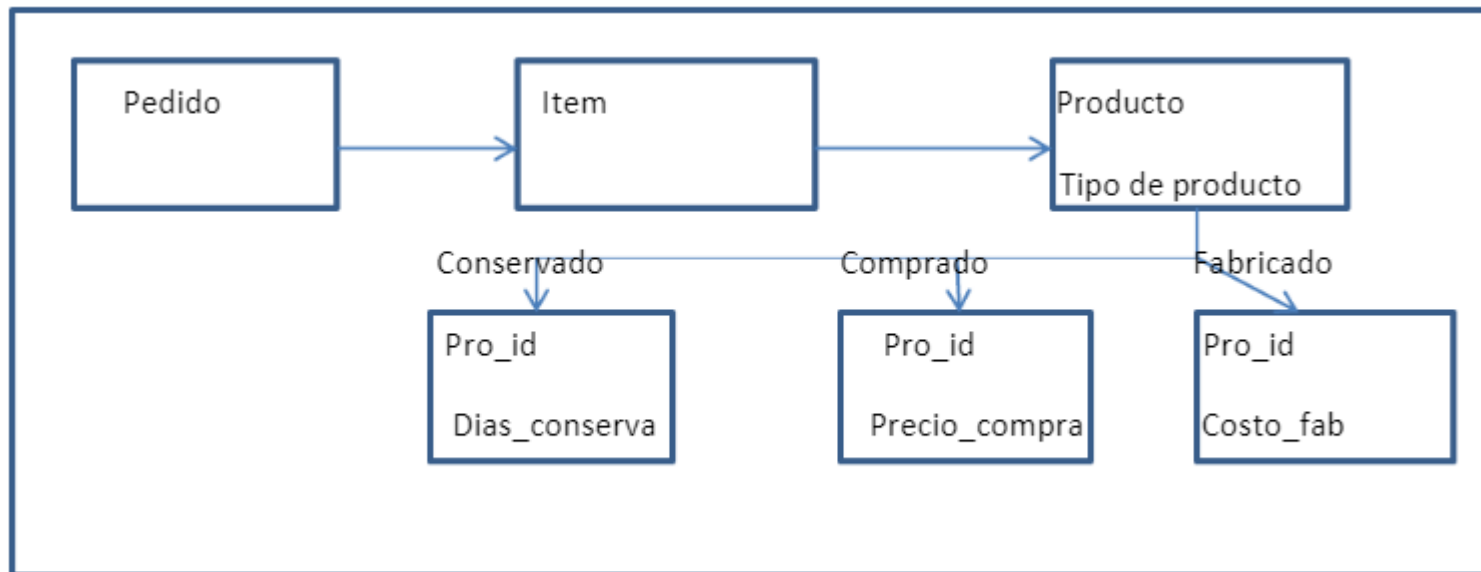
Todos estos campos constituirán la clave primaria de AxB

Relaciones de Herencia



Podemos establecer una equivalencia entre Tabla/Clase, y Registro/Instancias
Pero no siempre es tan lineal este mapping ...

Modelo Relacional



Modelo de Objetos

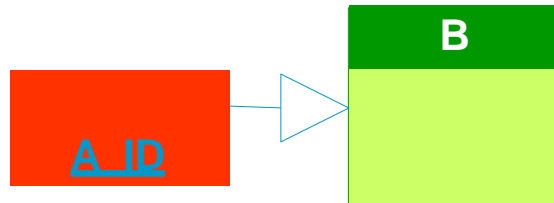


Al implementar este modelo tenemos 3 opciones:

- Implementar una tabla por cada clase
- Implementar una tabla por subclase
- Implementar una única tabla

Concepto	A favor	En Contra
Tabla	Facil. Mayor Performance. Evita generar muchas tablas	Campos no utilizados
Tabla por cada Clase	Es el modelo “ideal” según las reglas de Normalizacion Permite trabajar con queries polimórficas(no se necesita saber en que tabla esta la información) Permite campos no nulos para cada subclase	Es la opción que mas tablas crea
Tabla por Subclase	Permite establece campos no nulos y no requeridos	Cada subclase repite campos heredados

Algoritmo IV



Para cada relación de herencia

Sea C la superclase, con atributos $\{k, At_1, At_2, At_n\}$, con k marcado como clave y $\{S_1, S_2, \dots S_k\}$ las subclases de C

Ejecutar

Opción1

☐

Opción2

☐

Opción3

☐

Opción4

Algoritmo V

Opción1 Una tabla para cada clase

Crear una tabla tC para C , con atributos $\{k, At_1, At_2, At_n\}$.
Poner a k como Primary Key de tC

Para cada una de las subclases $\{S_1, S_2, \dots S_k\}$ Crear una
tabla tSi

Poner los atributos la clase Si como campos en tSi .

Poner a k simultáneamente como Primary Key de tSi y
como Foreign Key a la tabla tC

Fin Para

Algoritmo VI

Opción2

Una tabla para cada subclase

Para cada una de las subclases $\{S_1, S_2, \dots S_k\}$

Crear una tabla tSi

Poner los atributos de la clase Si como campos en tSi.

Poner los atributos de la clase C como campos en tSi.

Poner a k como Primary Key de tSi Fin Para

Algoritmo VII

Opción3 **Una tabla sola, con un campo para el tipo**

Crear una tabla tC

Poner los atributos de la clase C como campos en tC. Poner a k como Primary Key de tC

Para cada una de las subclases $\{S_1, S_2, \dots S_k\}$

Poner los atributos de la clase Si como campos en tC

Fin Para

Poner un campo T que indicará la subclase a la que pertenece el registro

Algoritmo VIII

Opción4 **Una tabla sola, con varios campos para el tipo**

Crear una tabla tC

Poner los atributos de la clase C como campos en tSi.

Poner a k como Primary Key de tC

Para cada una de las subclases $\{S_1, S_2, \dots S_k\}$

Poner los atributos de la clase Si como campos en tC

Fin Para

Poner un campo ti booleano, para subclase Si. Un valor true en ti indicará que el registro pertenece a la subclase ti

JPA Test Drive I

Problema

Quiero guardar instancias de esta clase en una base de datos

Book.java

```
public class Book {  
    private String isbn;  
    private String name;  
    private Publisher publisher;  
    private Date publishDate;  
    private int price;  
    private List chapters;  
  
    // Getters and Setters  
}
```

JPA Test Drive II

Solución

Incluir los jars de JPA en el proyecto



JPA Test Drive III

2

Configurar annotations en la clase

```
import javax.persistence.Column; import  
javax.persistence.*;
```

```
@Entity
```

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

```
public class Book {
```

```
    @Column (name="isbn")
```

```
    @Id
```

```
    String isbn;
```

```
    @Column (name="book_Name")
```

```
    String bookName;
```

```
    @Column (name="publisher_code")
```

```
    String publisherCode;
```

```
    @Column (name="publish_date")
```

```
    Date publishDate;
```

```
    @Column (name="price")
```

```
    Long price;
```

```
    //Getters y Setters
```

```
}
```

JPA Test Drive IV

Crear un archivo de configuracion para la persistencia

3

persistence.xml (en el directorio src/META-INF)

```
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
    http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
  Version="2.1">
  <persistence-unit name="ejshibernate" transaction-type="RESOURCE_LOCAL">
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>

  </persistence-unit>
</persistence>
```

JPA Test Drive V

Persistir y recuperar la instancia !

4

```
EntityManagerFactory managerFactory = Persistence
    .createEntityManagerFactory("ejshibernate");

EntityManager em = managerFactory.createEntityManager();

Book libro = new Book();

Libro.set().... ;

en.persist(libro);
```


JPA Test Drive VI

Persistir y recuperar la instancia !

4

```
Book libro2 = em.find(Book.class, isbn);
```

```
List libros = em.createQuery("select b from Book b").getResultList();
```

```
Object unlibro = m.createQuery("select b from Book b").getSingleResult();
```

Ejercicio



Ejercicio III de la guía



The background is a solid blue color with a subtle gradient. A wavy line, resembling a horizon or a stylized wave, separates the upper and lower portions of the image. The line starts flat on the left and curves upwards towards the right.

Muchas gracias!

TOGETHER. FREE YOUR ENERGIES