E.T.S. INGENIERÍA INFORMÁTICA

Gestión de la Información

UNIVERSIDAD DE MÁLAGA

LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA

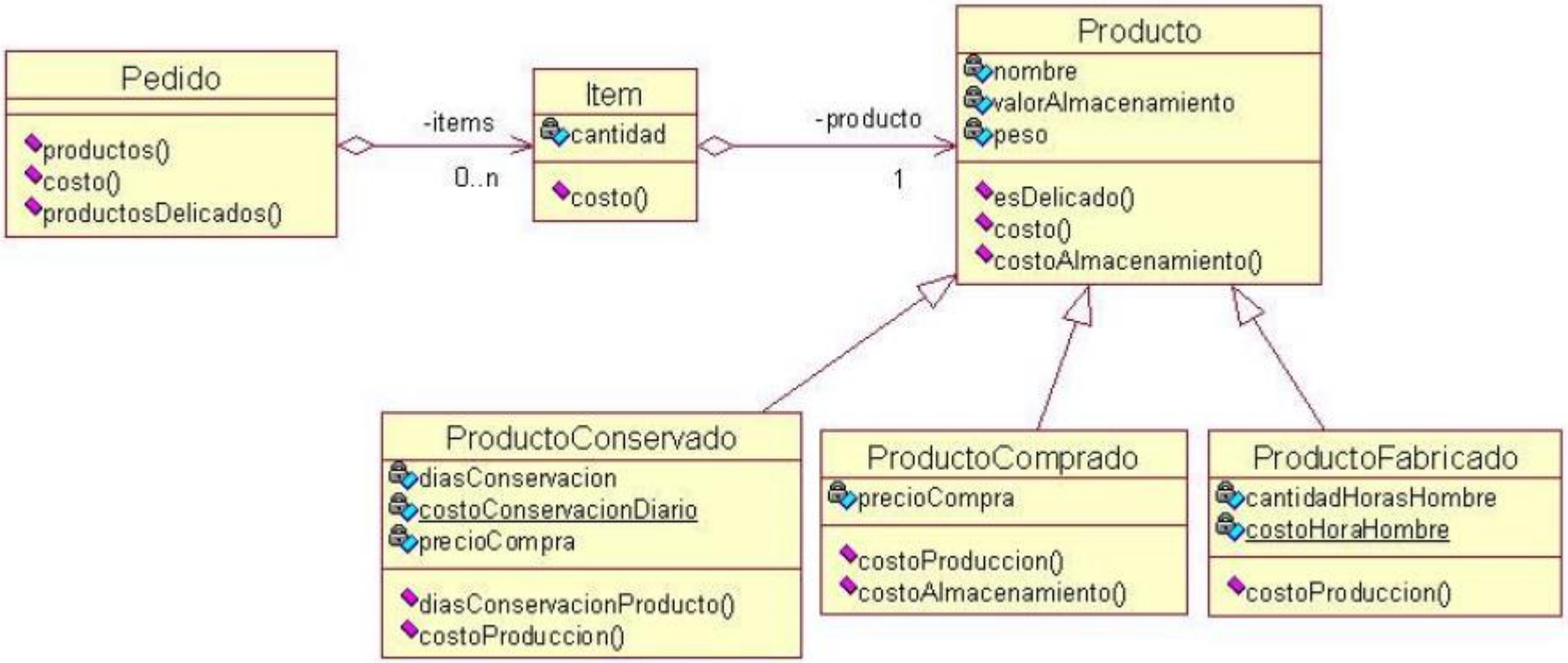
Tema 2

Transformaciones entre el modelo Relacional y el modelo de Clases.

José Luis Pastrana Brincones (pastrana@lcc.uma.es)

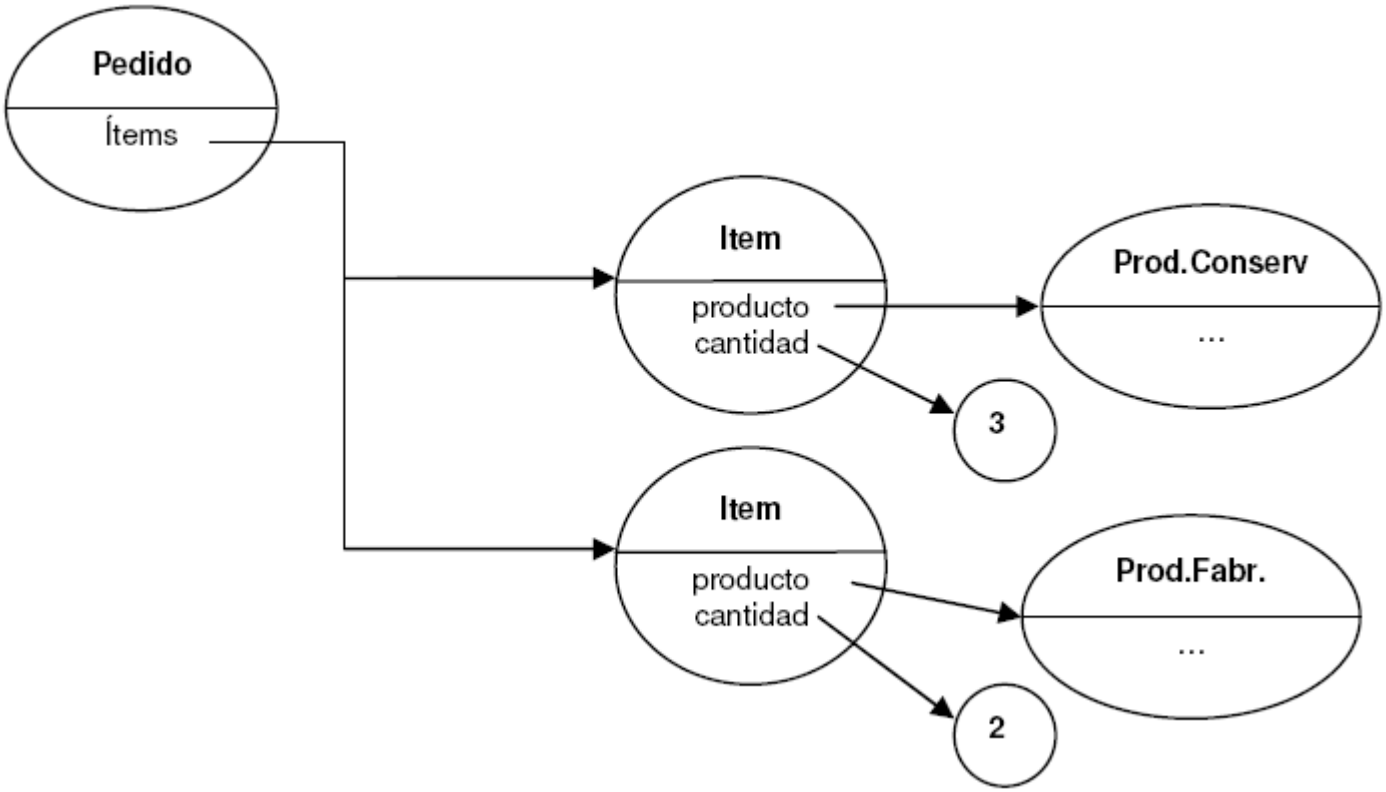
Introducción

- Supongamos que tenemos un modelo de objetos como el siguiente:



Introducción

- ¿Qué tipo de estructura siguen los objetos en memoria? ¿Cómo están almacenados?



Introducción

- La memoria es limitada y necesito poder almacenar los objetos en un medio que me permita recuperarlos (persistencia).
- La persistencia de la información es la parte más crítica en una aplicación de software.
- Opciones:
  - Puedo conservar la misma estructura en una base de objetos
  - Puedo utilizar una base de datos relacional como repositorio de información.

Introducción

- Si la aplicación está diseñada con orientación a objetos, la persistencia se logra mediante:
  - serialización del objeto (por ejemplo en XML)
  - o almacenando en una base de datos.

- El modelo de objetos difiere en muchos aspectos del modelo relacional.

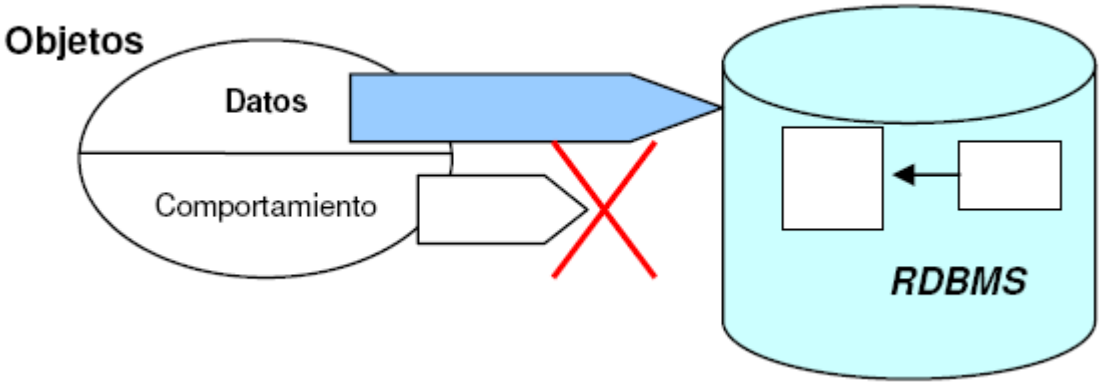
- La interface que une esos dos modelos se llama marco de Mapeo objeto-relacional (ORM en inglés).

Mapeo objeto-relacional

- Para persistir un objeto, normalmente:
  - se crea una conexión a la base de datos
  - se crea una sentencia SQL
  - Se asignan los parámetros
  - se ejecuta la transacción.
- Imaginemos que tenemos un objeto con varias propiedades, además de varias relaciones,
- ¿como las asociamos relacionalmente?
- ¿Cómo las almacenamos? ¿Automáticamente, manualmente? ¿Qué pasa con las claves?

Mapeo objeto-relacional

- Inicialmente podemos establecer una equivalencia entre el concepto de Tabla/Clase, y Registro/instancias de esa clase.



- Pero más adelante empezaremos a tener ciertas dificultades para que este mapping sea tan lineal.

Mapeo objeto-relacional

Objetos

- Tienen comportamiento.
- Los objetos encapsulan información para favorecer la abstracción del observador.
- Puedo generar interfaces.

Tablas

- controles de integridad o pequeñas validaciones (constraints o triggers).
- Una tabla no tiene esa habilidad.
- En el álgebra relacional la interfaz no se convierte en ningunaentidad

Mapeo objeto-relacional

- Los *Procedimientos Almacenados* no están asociados a una tabla, por lo que si toco un campo y necesito al menos recompilar todos los *Procedimientos Almacenados* asociados.
- La herencia es una relación estática que se da entre clases, que favorece agrupar comportamiento y atributos en común, pero en la implementación física las tablas no tienen el concepto de herencia.

Mapeo objeto-relacional

- En el álgebra relacional no hay polimorfismo ni vinculación dinámica.
- Al realizar una consulta sobre una tabla, necesito saber de qué tabla se trata.
- A todo esto es lo que se conoce como "*Impedance mismatch*" : las dificultades de traducción entre los dos mundos.
- Estamos usando la base de datos relacional sólo como repositorio para almacenar los datos de los objetos.
- El comportamiento sigue estando en los objetos cuando se instancian. Por ese motivo tenemos la posibilidad de adaptar el modelo relacional al de objetos

Mapeo objeto-relacional

- Hace algunos años teníamos que hacer manualmente ese mapeo objetos-relacional.
- Para persistir un objeto había que hacer un INSERT o un UPDATE
- Cuando queríamos recuperar un objeto había que hacer un SELECT de la tabla de clientes, y a partir de ahí por cada registro teníamos que construir el objeto.
- Hoy tenemos varios frameworks que manejan el mapeo Objetos-Relacional (OR/M Object Relational Mapping): Hibernate, OJB, JDO, LINQ, etc, que se han transformado en estándares de facto.
- La existencia de estos frameworks facilitaron la aceptación por parte del mercado de lo que tenemos hoy en día: reglas de negocio implementadas en una tecnología orientada a objetos + un framework de persistencia OO contra una base de datos relacional.

Ejemplo Mapeo objeto-relacional

- Supongamos que un Pedido tiene una fecha de pedido. Creamos la entidad Pedido:

Modelo relacional: Tabla Pedido

PEDIDO

PEDIDO_ID
FECHA_PEDIDO

Modelo de objetos: Clase Pedido

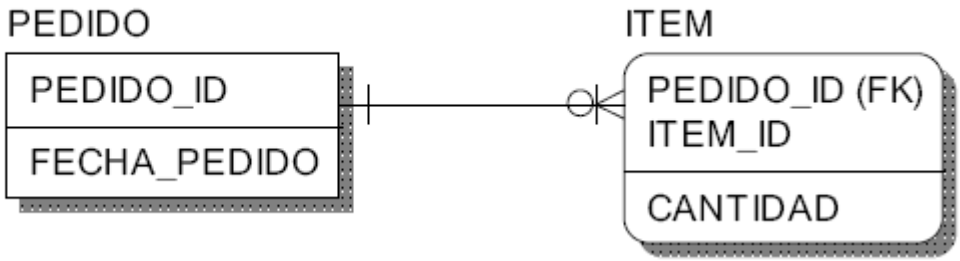
PedIdo
-fecha

- En el modelo relacional necesitamos una clave que identifique unívocamente a cada registro de la tabla Pedido. Estamos hablando del mismo registro si tienen el mismo PEDIDO_ID.
- En POO manejamos el concepto de identidad: Cuando yo referencio a un objeto no necesito identificarlo Hablamos del mismo objeto si objeto1 == objeto2.

Ejemplo Mapeo objeto-relacional

- Tenemos que manejar la relación Pedidos-Items-Productos.

Modelo relacional:

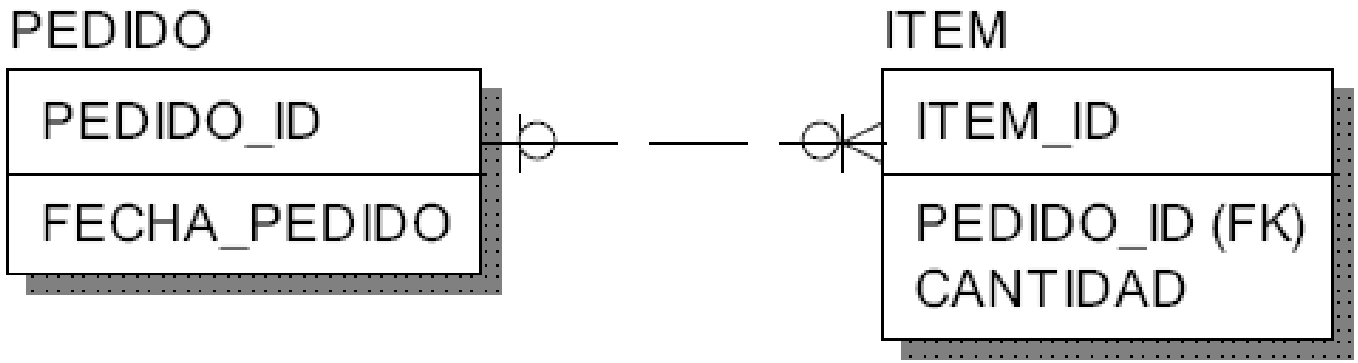


- Como hay una relación de 1 a muchos la clave principal de ITEM se compone del Identificador del Pedido (FK) y el del Identificador del Item.

Pedido_Id	Item_Id	Cantidad
1	1	...
1	2	...
2	1	...
2	2	...

Ejemplo Mapeo objeto-relacional

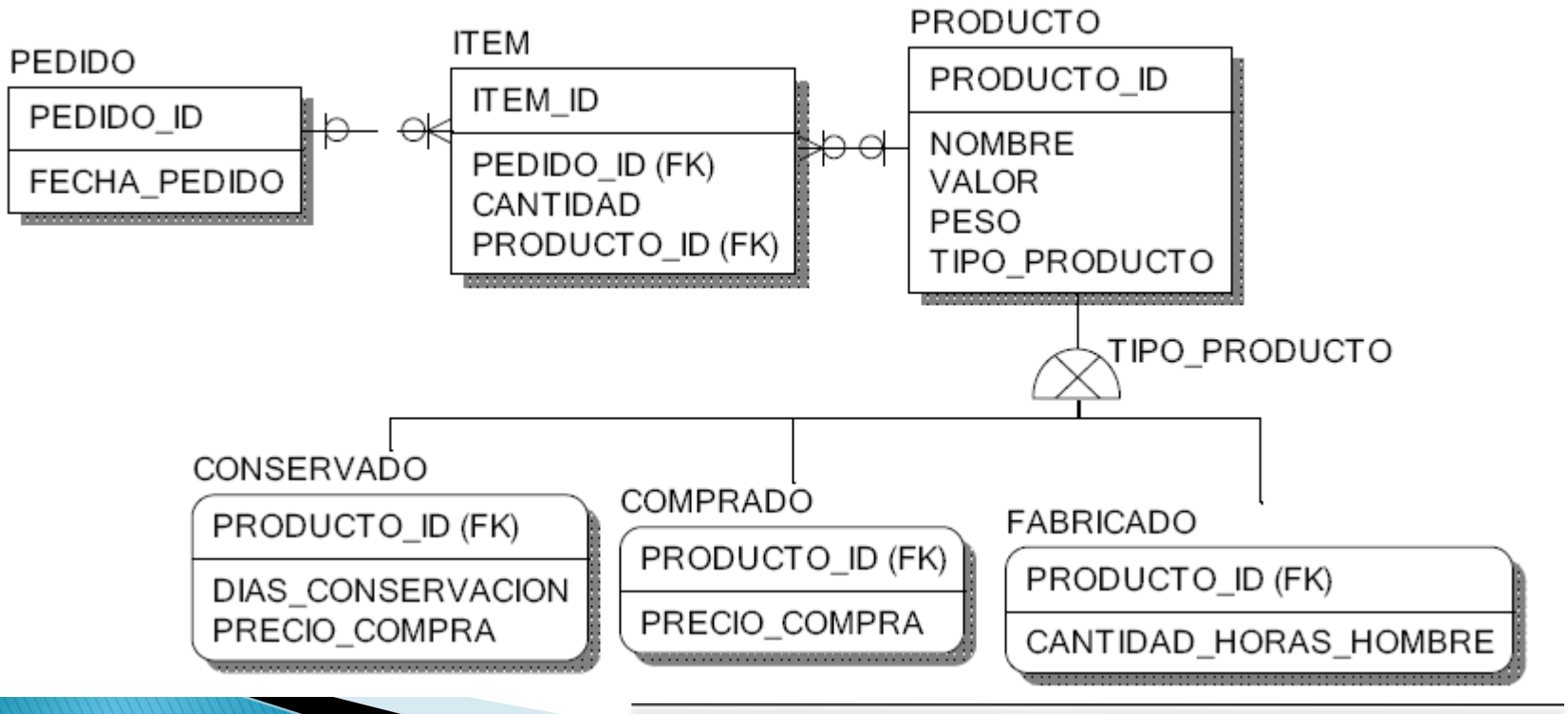
- Otra opción es definir al ítem como autoincremental



Item_Id	Pedido_Id	Cantidad
1	1	...
2	1	...
3	2	...
4	2	...

Ejemplo Mapeo objeto-relacional

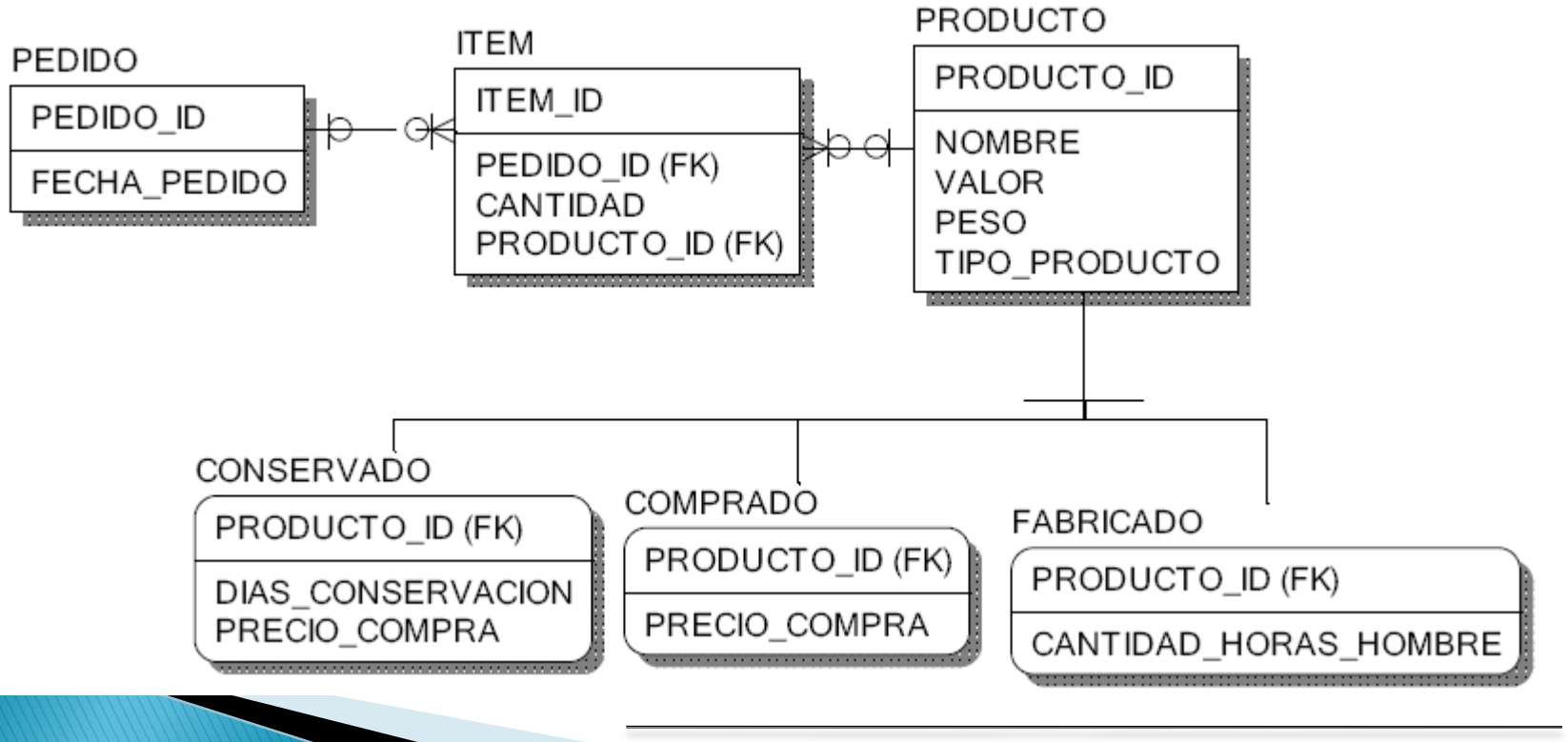
- Vamos a ver la entidad Producto.
- El modelo lógico equivalente a la clase Producto con sus subclases sería:



Ejemplo Mapeo objeto-relacional

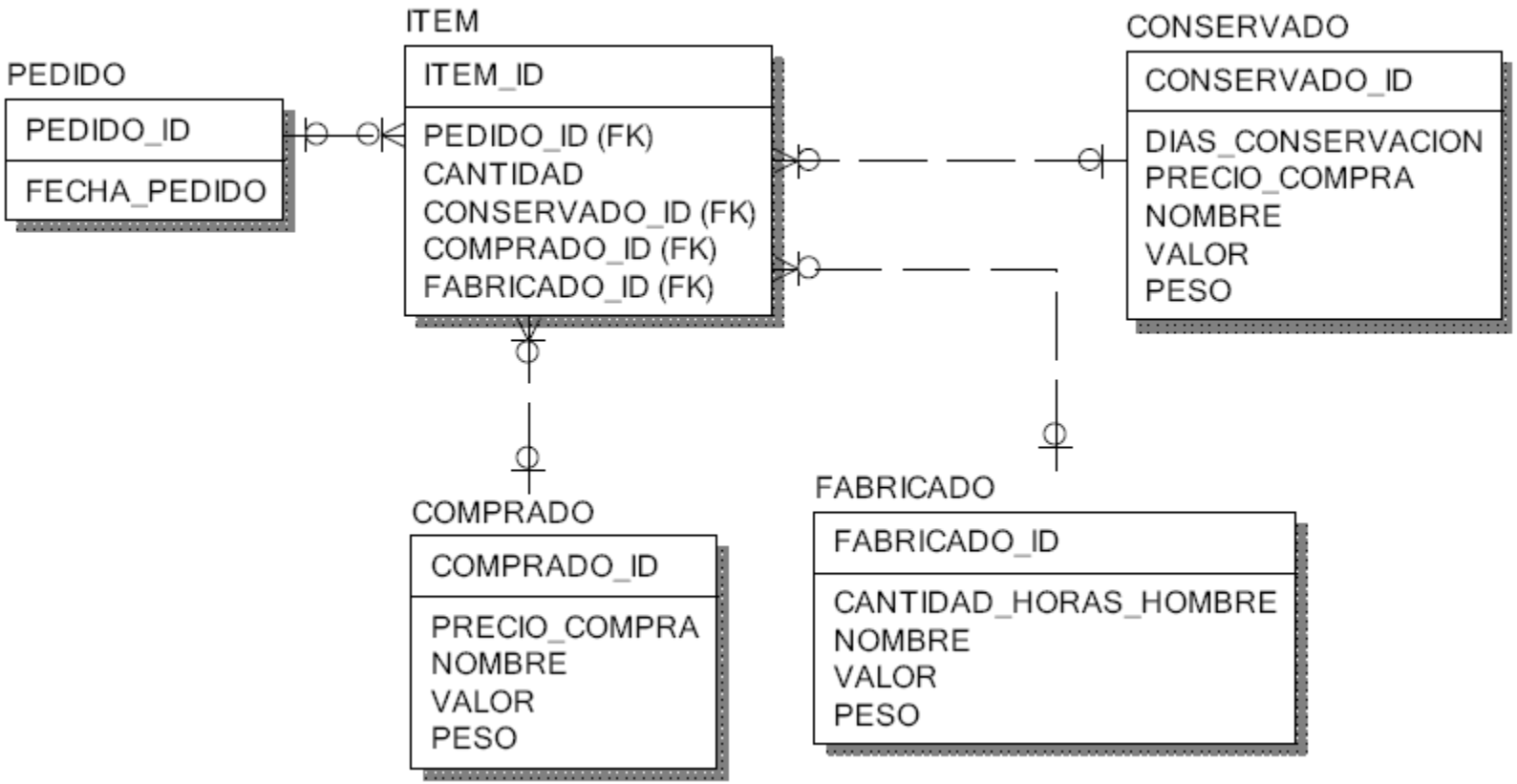
- Al implementar físicamente este modelo podemos elegir 3 opciones:

1. Implementar una tabla por cada clase:



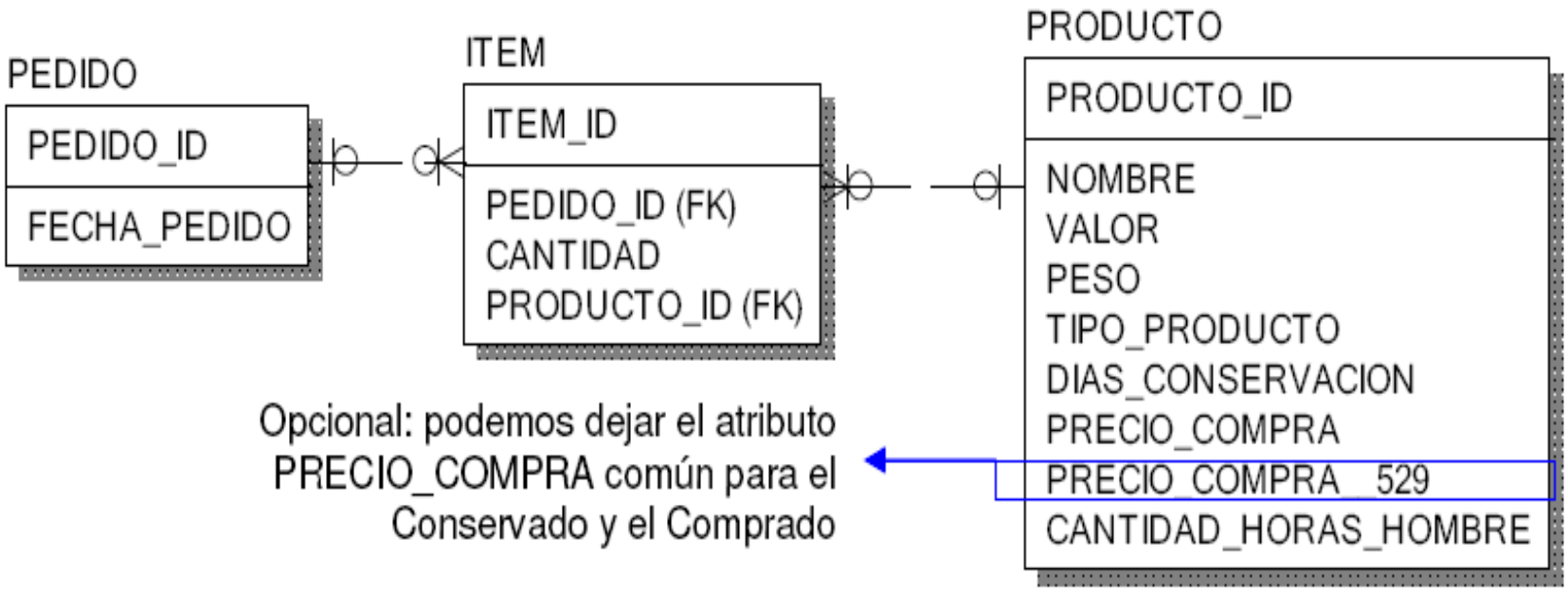
Ejemplo Mapeo objeto-relacional

2. Implementar una tabla por subclase:



Ejemplo Mapeo objeto-relacional

3. Implementar una única tabla:



Revisión Mapeo objeto-relacional

Solución	A favor	En contra
1 tabla	• Facilidad/Performance • Evito generar muchas tablas	• Campos no utilizados • Tentación de “reutilizar” atributos para cosas distintas
1 tabla por cada clase (n + 1)	• Es el modelo “ideal” según las reglas de normalización • No necesito saber en qué tabla está la información. • Permite establecer campos no nulos para cada subclase	• Es la opción que más entidades requiere crear • Requieren hacer LEFT JOIN contra todas las tablas que representan las subclases
1 tabla por subclase (n)	• Permite establecer campos no nulos para cada subclase • No requiere tener un campo discriminador (TIPO_PEDIDO)	• Cada subclase repite atributos “heredados” de la superclase

Revisión Mapeo objeto-relacional

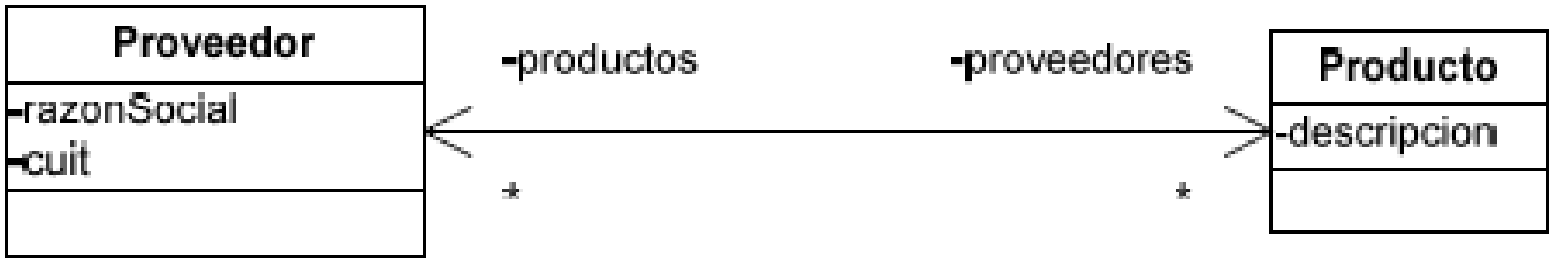
Solución	Cuándo conviene
1 tabla	• Cuando las subclases comparten muchos atributos en común. • Cuando se necesita simplificar las consultas.
1 tabla por cada clase (n + 1)	• Cuando hay muchos atributos comunes entre las subclases pero también muchos atributos propios en cada subclase.
1 tabla por subclase (n)	• Es la técnica utilizada para las entidades independientes. • Cuando las subclases comparten muy pocos atributos entre sí • Cuando no me interesa trabajar con consultas polimórficas (trabajo en forma independiente cada subclase)

Relaciones

Relaciones muchos a muchos

- Ejemplo 1: “Un proveedor vende muchos productos y un producto es vendido por muchos proveedores”.
- ¿Cómo lo modelo en objetos? Son colecciones desde los objetos raíces (proveedor y producto respectivamente).

Diagrama de clases



Navegación entre objetos

- El grafo de objetos puede navegarse en cualquier sentido porque todos los objetos están en memoria.
- Tengo un cliente, le pido el total y el cliente tiene acceso a sus pedidos. Para conocer el importe total, cada pedido tiene acceso a sus ítems, y cada ítem conoce su cantidad y tiene una referencia al producto.
- Si el Pedido tiene Items y cada Item tiene Producto, por un lado es cómodo traer toda la estructura del objeto Pedido para poder navegarlo libremente.
- Por otro lado, quizás arme toda una estructura sólo para averiguar la fecha de un pedido.

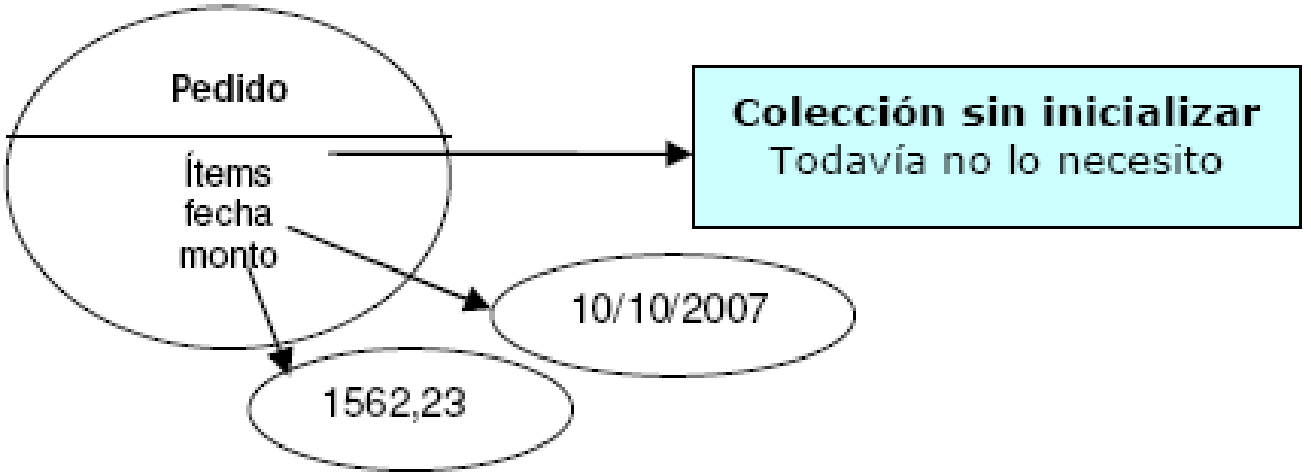
Navegación entre objetos

1. Obtengo sólo el pedido.

Queremos conocer todos los pedidos de un cliente:

```
SELECT * FROM PEDIDO WHERE CLIENTE_ID = 122
```

Esto se transforma en un objeto Pedido:



Navegación entre objetos

2. Obtengo el pedido y todas las relaciones asociadas:

Ahora tenemos que recuperar los pedidos del cliente:

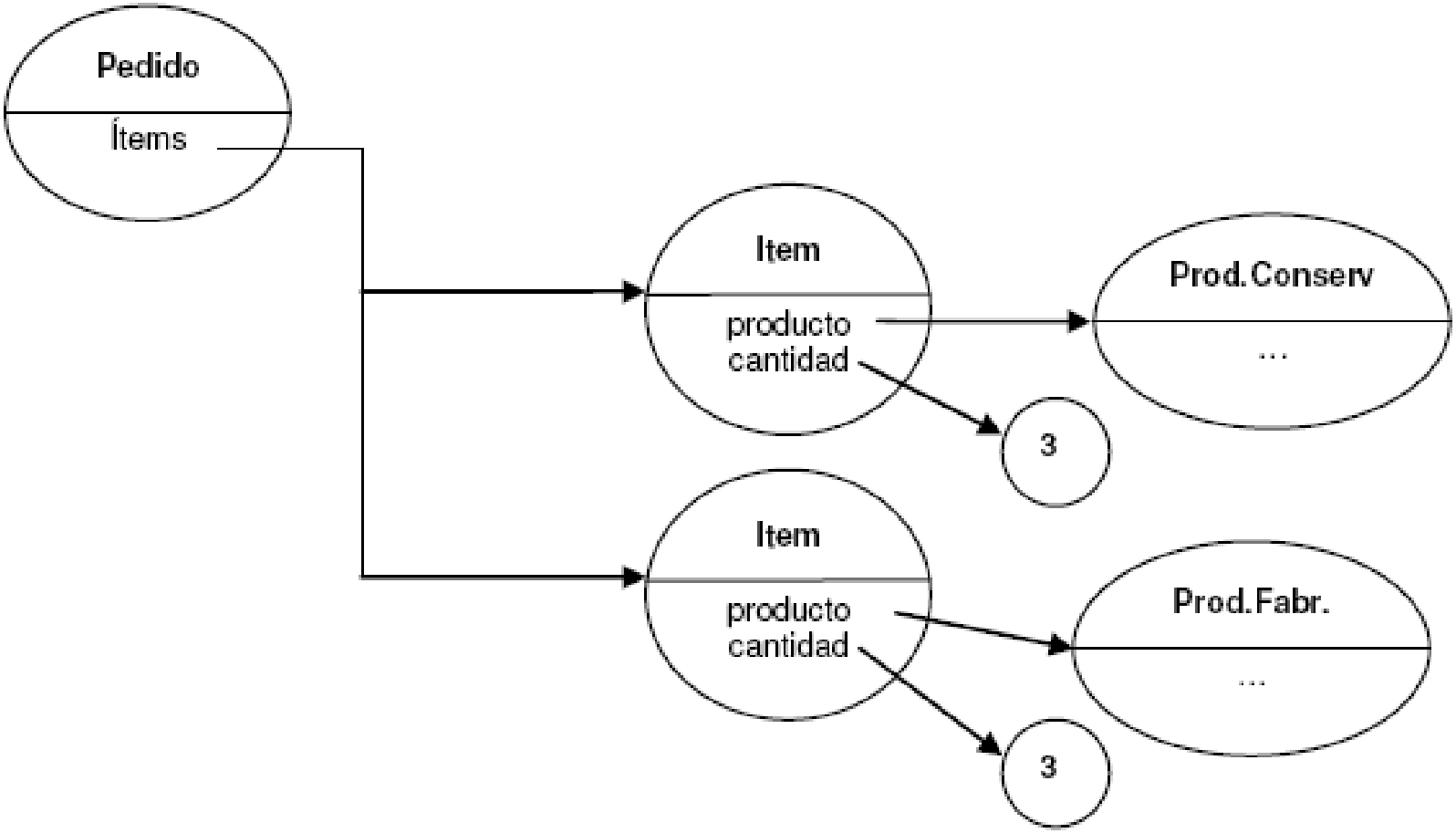
```
SELECT * FROM PEDIDO WHERE CLIENTE_ID = 122
```

Pero también tenemos que generar los ítems y los productos. Por cada ID_PEDIDO obtenido en la consulta anterior hacemos:

```
SELECT * FROM ITEM i INNER JOIN PRODUCTO p
ON p.ID_PRODUCTO = i.ID_PRODUCTO
WHERE i.ID_PEDIDO = ?
```

Navegación entre objetos

Con esto generamos:



Discusión

- ¿Me conviene (1) o (2)?
- En general los frameworks que hacen mapeo O/R ofrecen la posibilidad de definir hasta dónde voy a convertir.
- **Lazy association**: una asociación lazy, es aquella donde voy a traer la información bajo demanda (“sólo cuando lo necesite”).

Discusión ¿O/R vs R/O?

- Una vez adoptada la decisión de trabajar con bases relacionales, la pregunta es qué hacer primero:

a) a) Generamos el modelo relacional y luego adaptamos el modelo de objetos en base a las tablas generadas

b) b) Generamos el modelo de objetos y en base a éste se crean las tablas.

- La opción a) supone que es más importante la forma en que guardo los datos que las reglas de negocio que modifican esos datos.
- En la opción b) el desarrollo con objetos no se ve ensuciado por restricciones propias de otra tecnología.