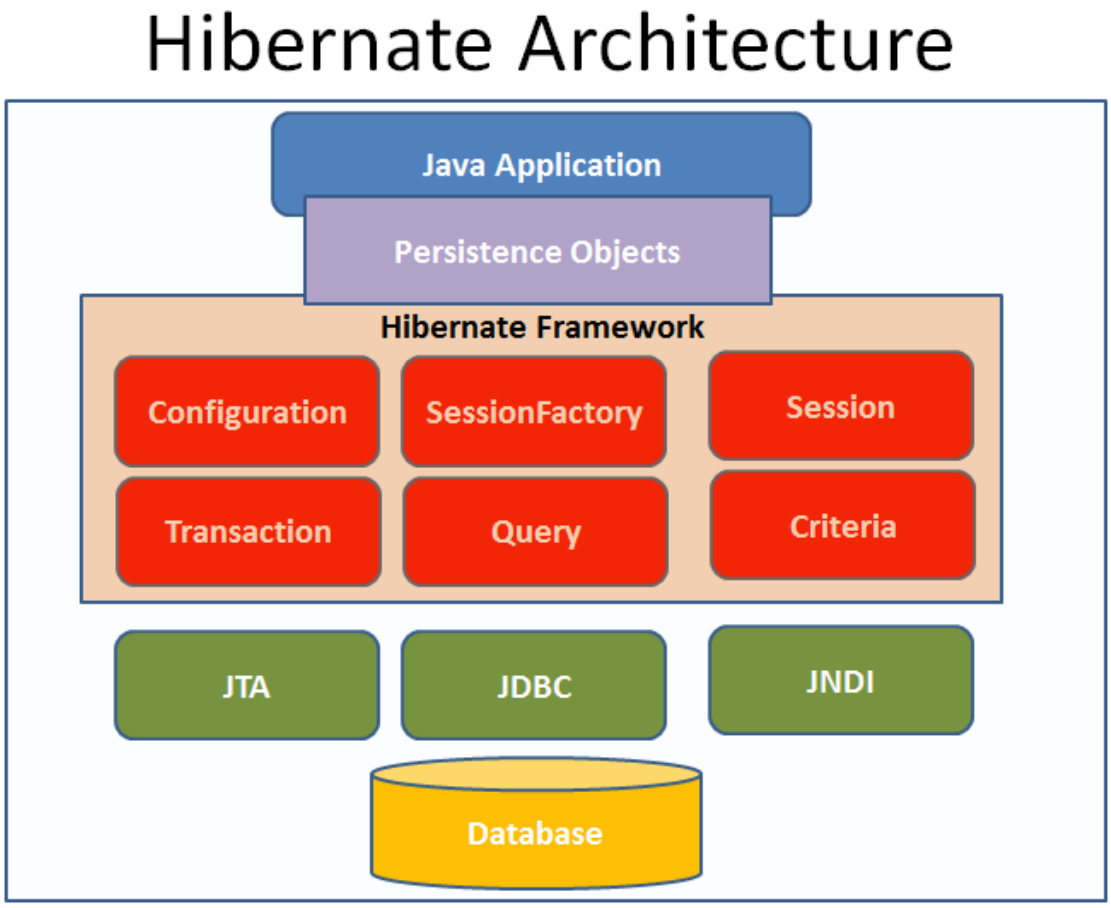
Práctica 1: ORM y Hibernate



Alumnes:
Nombre: Michel Dennis Quitaquis
NIUB: 14991045

Nombre:Tian Lan
NIUB: 16129094

ÍNDICE:

INTRODUCCIÓN

En esta práctica, vamos a realizar una aplicación JAVA sobre venta de artículos. La idea es aplicar el framework de persistencia (ORM, Hibernate) para gestionar los objetos persistentes, así nos permite trabajar con nuestra base de datos a nivel de JAVA y evitaremos tener que trabajar a nivel de SQL, la aplicación estará independiente del SGDB. Para simplificar, vamos a utilizar el SQLITE para la creación de la base de datos (TABLES) en vez de conectarse a un servidor.

OBJETIVOS

-Entender mejor que es una ORM.
-Familiarizarse con el Hibernate (HQL, Mappings...)
-Conocer los diferentes aspectos del Hibernate (estrategia de carga, herencia…)
-Integrar el uso de la base de datos en la aplicación.

DIAGRAMAS DE MODELO ER Y RELACIONAL

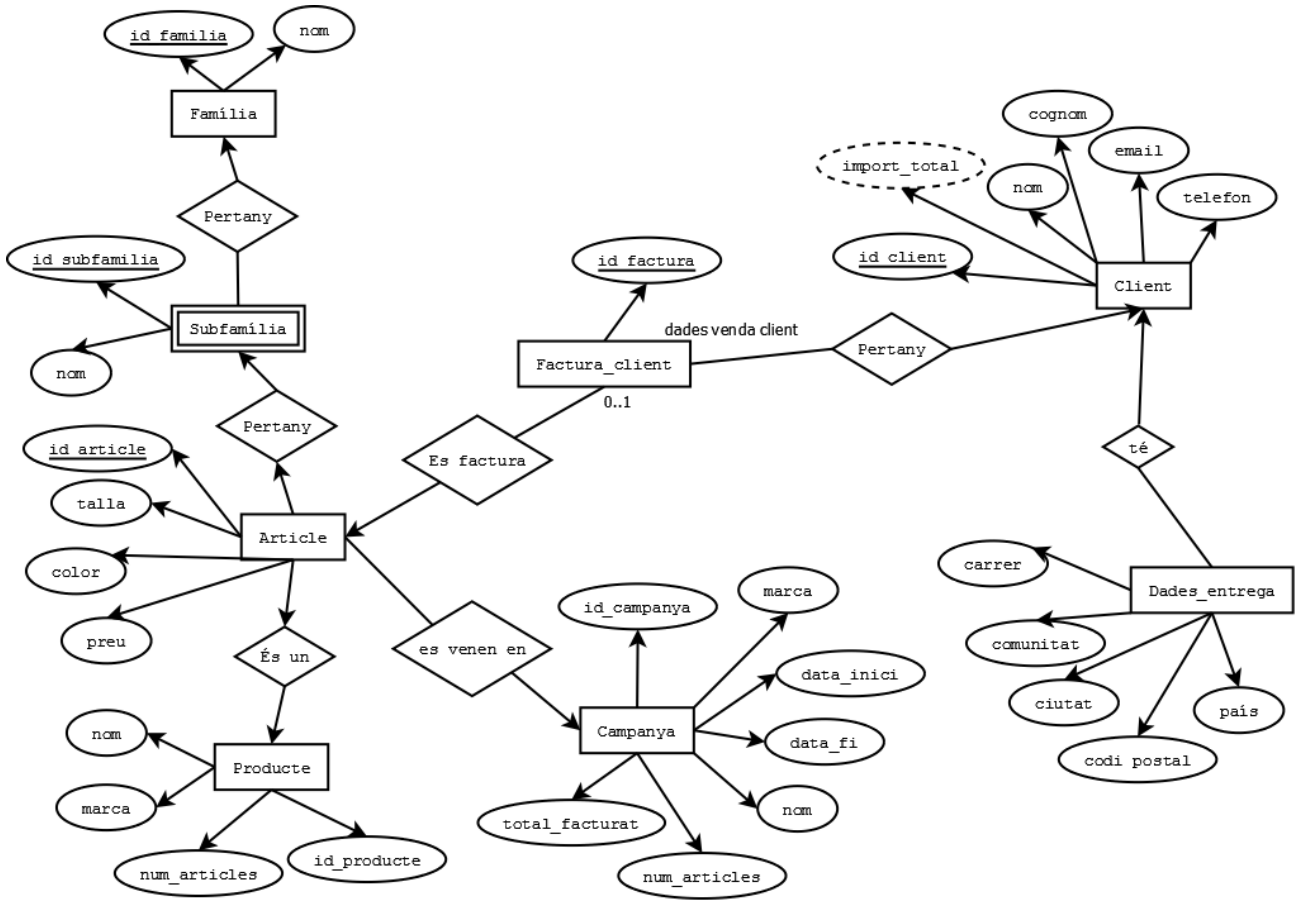


fig 1: modelo E-R de la nuestra base de datos

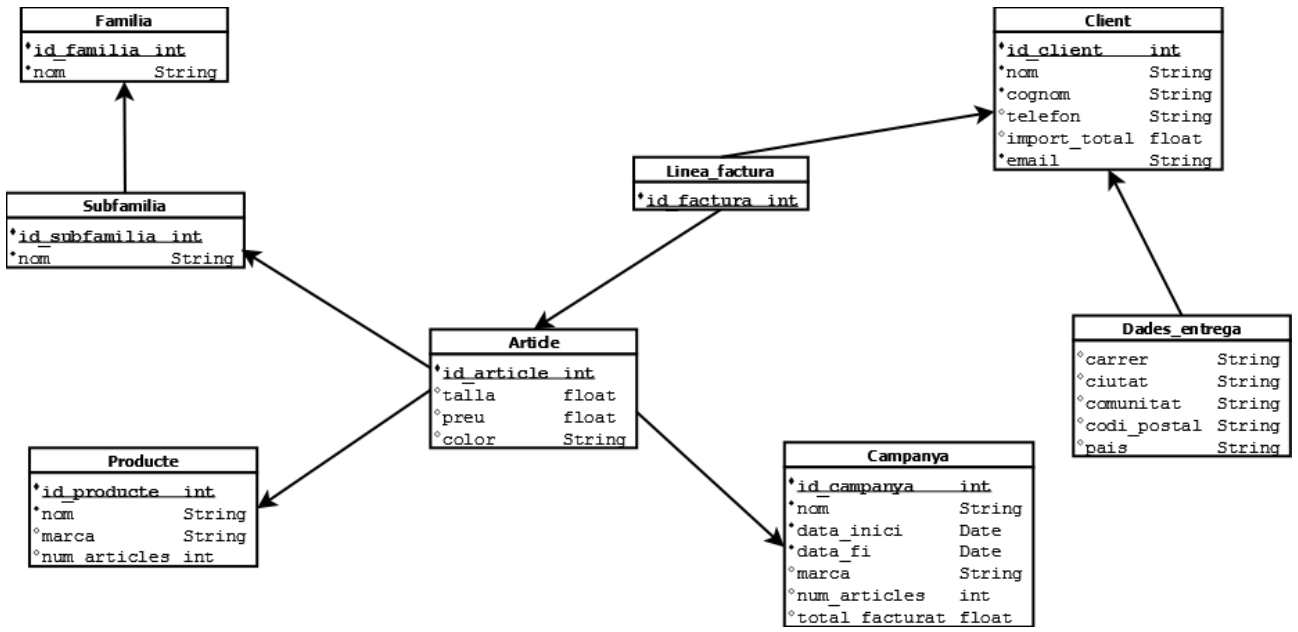


fig 2: modelo relacional de la nuestra base de datos

DESARROLLO

Base de datos (con SQLITE):

Hemos creado una tabla para cada una de las entidades que se muestran en los diagramas, especificando los PRIMARY KEYS, FOREIGN KEYS y los atributos con sus dominios. Luego, hemos creado una base de datos pequeña con el excel y luego lo hemos exportado a 'CSV' para la inserción automática de la misma.

Esquema de las tablas y explicación de algún detalle:

Familia(id_familia:int, nom:String)
El atributo "nom" puede ser {hombre, mujer}.

Subfamilia(id_subfamilia:int, nom:String, id_fam:int),dominio={
El atributo "nom" puede ser {camisetas, camisas, chaquetas, abrigos, vaqueros, pantalones, ropa_interior, calcetines} para hombres y {vestidos, chaquetas, abrigos, pantalones, faldas, ropa_interior, calcetines, vaqueros} para mujeres.

Producte(id_producte:int, nom:String, marca:String, #articles:int)
Un producto guarda el número de artículos que son de este producto. De ser así, siempre que se venda un artículo, este valor se resta en 1.

Article(id_article:int, talla:float, color:String, preu:float, id_subfam:int, id_campanya:int, id_producte:int)
Relacionado con producto, habrá artículos que son del mismo producto, pero cada uno tiene sus propias características diferenciables.

Client(email:String, nom:String, cognom:String, telefon:String, import_total:float)
import_total es el total que ha gastado un cliente en las campañas.

Linea_factura(id_factura:int, id_client:String, id_article:int)
Cada tupla de "Linea_factura" está relacionada con 1 tupla de "Article" y pertenece a un cliente. Eso quiere decir que cada tupla representa una línea de venta, y un cliente tiene tantas líneas de venta como #artículos que ha comprado.

Dades_entrega(id_dades_entrega:int, carrer:String, comunitat:String, ciutat:String, codi_postal:String, pais:String, id_cliente:String)
Esta entidad representa a la dirección física de un cliente.

Campanya(id_campanya:int, nom:String, marca:String, #articles:int, total_facturat:float, data_inici:Date, data_fi:Date)

Aplicación

Modelo: aquí están las clases .java y los ficheros .xml que mapean estas clases con las tablas de base de datos. Las relaciones entre las tablas también son configuradas en los .xml y a través de poner objetos como atributos en la clases correspondientes.
-Un objeto de B en la clase A: si cada objeto A se relaciona con uno de B
-Una colección de objetos B en la clase A: si cada objeto A se relaciona con muchos de B.

También se hace el mapeo de los atributos a través del tag “property”, el atributo PK se configura con el tag “id”.
En cada clase se han creado un constructor sin parámetros, los setters y getters de todos los artículos.

Vista:
ConnectorHB: crea un objeto de “SessionFactory” y lo devuelve.
Menu: Controla las opciones del menú principal y de los submenús y se encarga de mostrarlas.
Privalia: Instancia al “Session Factory” y lanza el menú principal. Gestiona el flujo de menús y aquí están implementadas las operaciones CRUD para todas las tablas.

MANUAL DE LA APLICACIÓN

```
LOGIN
-----
User: admin
Pass: admin
```

fig 3: login screen

En esta aplicación sólo se admite un usuario (admin con la contraseña “admin”). En caso de introducir mal, se para la aplicación.

```
Instanciando SF
-----------------------------------------
MENU PRINCIPAL
-----------------------------------------
        1.- CRUD operations Familia
        2.- CRUD operations Subfamilia
        3.- CRUD operations Companya
        4.- CRUD operations Article
        5.- CRUD operations Client
        6.- CRUD operations Dades entrega
        7.- CRUD operations Linea factura
        8.- Sortir
-----------------------------------------
Entra una opcio >>
```

fig 4: menú principal de la aplicación

Vamos a seleccionar la opción 1. A continuación se muestra...

```
CRUD OPERATIONS FOR FAMILIA
-----------------------------------
        1.- Create
        2.- Read
        3.- Update
        4.- Delete
        5.- Tornar al menú anterior
-----------------------------------
Entra una opcio >>
```

fig 5: submenú para operaciones CRUD para familia

1.Create: permite crear un objeto de “Familia” introduciendo un nombre para él.
2.Read: Recuperar los objetos de “Familia” y los muestra por la consola.
3.Update: Actualizar el nombre de un objeto de “Familia” por el nombre introducido.
4.Delete: Eliminar un objeto de “Familia” indicando el id del mismo.
5.Volver al menú anterior.

El resto de submenús de operaciones CRUD funcionan de forma similar al que hemos visto anteriormente. Pues obviamos entrar en los detalles de la explicación de ellos.

PROBLEMAS SURGIDOS

En cuanto a la creación de la BD:
La entidad “Linea_factura”, discusión para saber qué entidad apuntaba a que. Client apuntaba a muchas facturas y “Article” es apuntado por muchas facturas.
No teníamos claro de la existencia de “producte” relacionado con “article”.

En cuanto a la implementación de la aplicación:
Respecto al tema de herencia hemos intentado hacer la relación de herencia entre “producte” y “article”, pero nosotros al relacionar “article” con su “subfamilia”, “campanya” y “Linea_factura”, necesitamos una PK de “article” y al hacer la herencia, “article” comparte PK con “producte” (clase padre). Y por tanto, no lo hemos implementado de esta manera. Lo hemos implementado de tal manera que “article” tiene un FK de “producte” y se relacionan así.

Por la cuestión del **tiempo**, la aplicación queda muy básica y con un código que se puede modularizar más ya que no hemos utilizado un controlador para poder separar las diferentes consultas de cada tabla y así no sobrecargar la clase del paquete “Vista” (privalia.java). Además, falta implementar el control de excepciones para los diferentes tipos de entrada.

Integrar, configurar Hibernate, SQLITE3 con Netbeans.

El fichero de configuración de Hibernate no otorga la posibilidad de configurar la URL de conexión de manera relativa, es decir, hay que introducir la URL física donde se encuentra la BD.

CONCLUSIÓN

Aún por la falta del tiempo, hemos podido observar que realizar consultas con Hibernate es más sencillo ya que nos permite enmascarar las consultas anidadas como si POO fuese, aunque el diseño de la BD y el mapeo de las clases con las tablas afecta mucho a la programación posterior.