BASES DE DATOS 2

MEMORIA DE LA PRÁCTICA 4
CURSO 2017-2018

CARLOS MARAÑES NUENO 717788@CELES.UNIZAR.ES

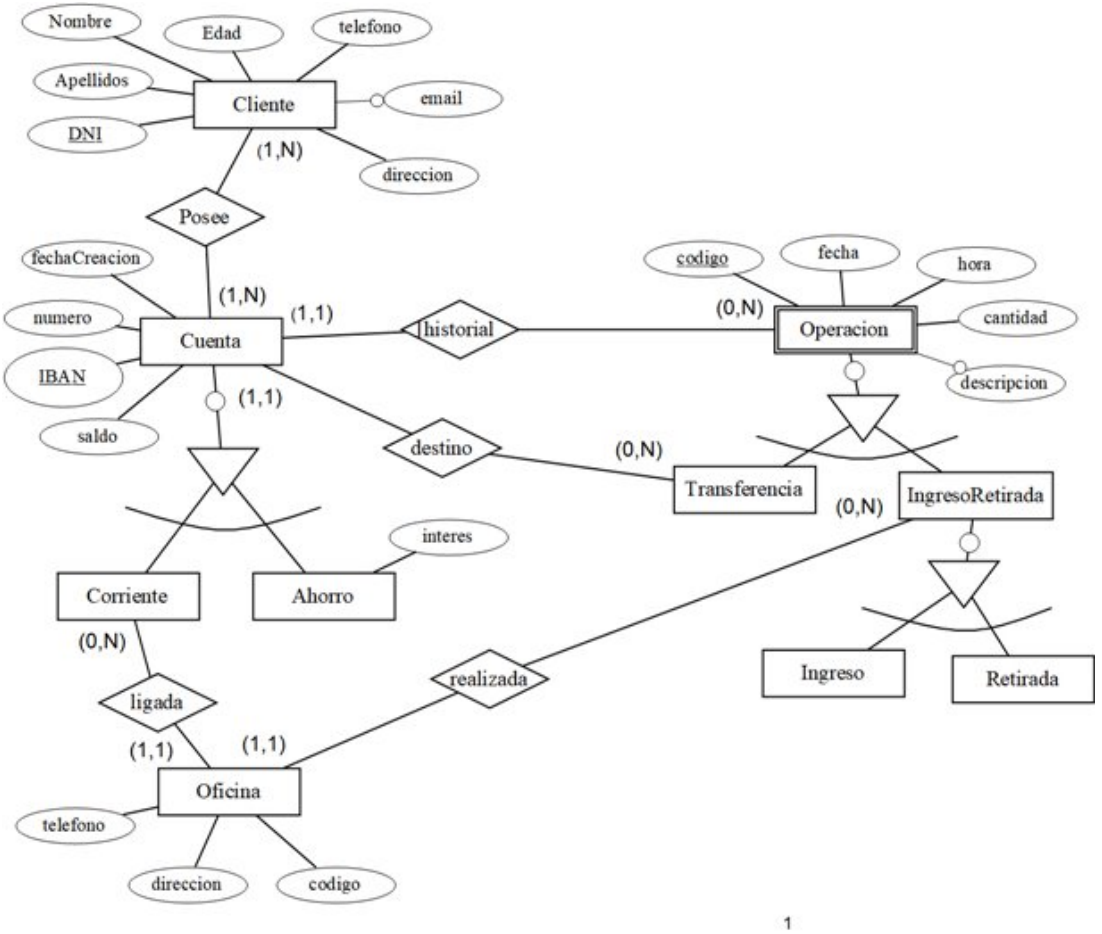
NICOLÁS LERA LÓPEZ 721808@CELES.UNIZAR.ES

ÍNDICE

1. INTRODUCCIÓN

El objetivo de esta práctica es practicar con la herramienta Hibernate, implementando en Java el código necesario para mapear las tablas creadas en la práctica 1 del problema del banco. Además, también se realizan consultas en los lenguajes SQL, JPQL y Criteria API.

2. DE JPA A ORACLE



1

La figura anterior es el diagrama E/R realizado en la práctica 1. En JPA se ha creado una clase por cada entidad. Todas las relaciones se han hecho bidireccionales. Las especializaciones de Cuenta y Operación se han resuelto en una sola tabla en la base de datos, para mejorar el rendimiento a la hora de consultar. Las sentencias SQL ejecutadas han sido las siguientes:

```
Hibernate: drop table H_CLIENTE cascade constraints
Hibernate: drop table H_CLIENTE_H_CUENTA cascade constraints
Hibernate: drop table H_CUENTA cascade constraints
Hibernate: drop table H_OFICINA cascade constraints
Hibernate: drop table H_OPERACION cascade constraints
```

```
Hibernate: create table H_CLIENTE (DNI varchar2(255 char) not null, APELLIDOS varchar2 (255 char), DIRECCION varchar2(255 char), EMAIL varchar2(255 char), FECHANAC date, NOMBRE varchar2(255 char), primary key (DNI ))
Hibernate: create table H_CLIENTE_H_CUENTA (clientes_DNI varchar2(255 char) not null , cuentas_IBAN varchar2(255 char) not null, primary key (clientes_DNI, cuentas_IBAN))
Hibernate: create table H_CUENTA ( DTYPE varchar2(31 char) not null, IBAN varchar2( 255 char) not null, CCC number(10,0), FECHA_CREACION date, SALDO number(10,0), INTERES double precision, oficina_CODIGO number(10, 0), primary key (IBAN))
Hibernate: create table H_OFICINA (CODIGO number(10,0) not null, DIRECCION varchar2(255 char), TELEFONO number(10,0), primary key (CODIGO))
Hibernate: create table H_OPERACION (DTYPE varchar2(31 char) not null, CODIGO number(10,0) not null, CANTIDAD double precision, DESCRIPCION varchar2(255 char), FECHA date, HORA varchar2(255 char), cuentaOrigen_IBAN varchar2(255 char), oficina_CODIGO number(10,0), cuenta_IBAN varchar2(255 char), primary key (CODIGO))
Hibernate: alter table H_CLIENTE_H_CUENTA add constraint FK3437FBD4C5181A foreign key (cuentas_IBAN) references H_CUENTA
Hibernate: alter table H_CLIENTE_H_CUENTA add constraint FK3437FB107AB85B foreign key (clientes_DNI ) references H_CLIENTE
Hibernate: alter table H_CUENTA add constraint FK4FE7A81FFFFC1620 foreign key (oficina_CODIGO) references H_OFICINA
Hibernate: alter table H_OPERACION add constraint FK90BE6BC12CA35CB3 foreign key (cuentaOrigen_IBAN) references H_CUENTA
Hibernate: alter table H_OPERACION add constraint FK90BE6BC1D044DB1D foreign key (cuenta_IBAN) references H_CUENTA
Hibernate: alter table H_OPERACION add constraint FK90BE6BC1FFFC1620 foreign key (oficina_CODIGO) references H_OFICINA
```

Cabe recalcar que el siguiente campo en el archivo persistance.xml tiene el siguiente valor:
`<property name="hibernate.hbm2ddl.auto" value="create"/>`
Ya que se crean las tablas se han de crear en Oracle.

La implementación de la relación entre Cliente y Cuenta se ha realizado en una sola tabla: H_CLIENTE_H_CUENTA. El resto de relaciones se han realizado mediante propagación de la clave en la entidad de mayor cardinalidad.

3. DE ORACLE A JPA

Se parte de las siguientes tablas en Oracle:

```
CREATE TABLE Cliente(
   DNI VARCHAR(10) NOT NULL,
   nombre VARCHAR(100) NOT NULL,
   apellido VARCHAR( 100) NOT NULL,
   edad INTEGER NOT NULL,
   Direccion VARCHAR(100) NOT NULL,
   email VARCHAR(100),
   Telefono VARCHAR(15) NOT NULL,
   PRIMARY KEY (DNI)
);

CREATE TABLE Oficina(
   codigo VARCHAR(15 ) NOT NULL,
   direccion VARCHAR (100) NOT NULL,
```

```
    telefono VARCHAR(  15) NOT NULL, /* mejor varchar para especificar formato */
    PRIMARY KEY (codigo)
);

CREATE TABLE Cuenta(
    IBAN VARCHAR(150) NOT NULL,
    numero INTEGER NOT NULL,
    saldo DECIMAL(10,   2) NOT NULL,
    fechaCreacion DATE NOT NULL,
    tipo VARCHAR(9) NOT NULL,
    interes DECIMAL(3    ,2),
    oficina VARCHAR(15),
    PRIMARY KEY(IBAN),
    FOREIGN KEY(oficina) REFERENCES   Oficina(codigo)
);

CREATE TABLE Posee(
    DNI VARCHAR(10) NOT NULL,
    IBAN VARCHAR(150) NOT NULL,
    PRIMARY KEY (DNI,   IBAN),
    FOREIGN KEY (DNI)   REFERENCES Cliente(DNI),
    FOREIGN KEY (IBAN  )    REFERENCES Cuenta(IBAN)
);

CREATE TABLE Operacion(
    codigo VARCHAR(150) NOT NULL,
    tipo VARCHAR(20) NOT NULL,
    fecha DATE NOT NULL,
    hora VARCHAR(8) NOT NULL,
    cantidad DECIMAL(   10,2) NOT NULL,
    descripcion VARCHAR(400),
          IBANOrigen VARCHAR(150) NOT NULL,
    IBANDestino VARCHAR(150),
    oficina VARCHAR(15),
    PRIMARY KEY (codigo, IBANOrigen),
    FOREIGN KEY (IBANOrigen) REFERENCES Cuenta(IBAN),
    FOREIGN KEY (IBANDestino) REFERENCES Cuenta(IBAN),
    FOREIGN KEY (oficina) REFERENCES Oficina(codigo)
);
```

Se ha utilizado una herramienta de generación automática de tablas perteneciente al software IntelliJ Idea. Sin embargo, las relaciones obtenidas no son del todo precisas, al igual que las especializaciones, por lo que se ha tenido que rehacer dicha parte.

La relación entre Cliente y Cuenta, que se ha implementado como una tabla, se ha de especificar en el código mediante la cláusula @JoinColumn.

La entidad Operación, que es de tipo débil, tiene una clave compuesta, por lo que para poder implementarla se ha creado una clase que sea la combinación de ambas claves.

Cabe recalcar que el siguiente campo en el archivo persistance.xml tiene el siguiente valor:
`<property name="hibernate.hbm2ddl.auto" value="validate"/>`

Ya que se crean las tablas se han de emparejar con las existentes.

4. CONSULTAS

Se han realizado 5 consultas sobre la base de datos en cada parte de la práctica. En ambas prácticas se han hecho prácticamente las mismas consultas. Entre estas se encuentran 3 realizadas en jpql, una en SQL nativo y otra en criteria api. A continuación se explican las consultas:

1. Alguna oficina donde tenga domiciliada una cuenta corriente el primer cliente alfabéticamente cuyo nombre tiene menos de 7 letras.

2. Cantidad de dinero media movida por oficina

3. Cuentas corrientes con saldo menor que diez y la oficina donde están domiciliadas

4. Transferencias recibidas por la cuenta de mayor saldo (Native Query)

5. Cuentas cuyas operaciones han sido con una cantidad entre 100 y 200 (Criteria api)

Código consultas parte 1:

```
//CONSULTA 0
String cabecera = "Consulta1:\n Alguna oficina donde tenga domiciliada"
        + "una cuenta corriente el primer\n cliente alfabeticamente cuyo"
        + " nombre tiene menos de 7 letras\n";
System.out.println(cabecera);
String q0 =  "select c FROM H_CLIENTE c where "
        + "LENGTH(c.nombre)<7 order by c.nombre";
Query query0 = em.createQuery(q0);
List<Cliente >res0 = query0.getResultList();
Cliente c0=res0.get(0 );
String q01 =  "select cc from H_CUENTA cc join cc.clientes "
        + "c where c.nombre = :nom";
Query query01 = em.createQuery(q01);
query01.setParameter("nom", c0.getNombre());
List<Cuenta>res01 = query01.getResultList  ();
Cuenta cc0=res01.get(0);
String q02 =  "select co FROM H_CORRIENTE co where co.IBAN = :ib";
Query query02 = em.createQuery(q02);
query02.setParameter("ib", cc0.getIBAN());
List<Corriente> res02 = query02.getResultList();
for(Corriente  co : res02){
        System.out.println(co.  getOficina().getDireccion()+"\n");
}


//CONSULTA 1
```

```
cabecera = "Consulta 2: Cantidad media movida por oficina\n";
System.out.println(cabecera);
String q1 =   "select o from H_OFICINA o";
Query query1 = em.createQuery(q1);
List<Oficina  >res1 = query1.getResultList();
System.out.println("OFICINA      Cantidad Media Movida");
for(Oficina o   : res1){
        Set<IngresoRetirada> l = o.getIngresoRetirada();
        int i   =0;
        double sumaSaldo=0;
        for( IngresoRetirada iR  : l){
                sumaSaldo += Math.abs(iR.getCantidad());
                i++;
        }
        sumaSaldo = sumaSaldo/i;
        System.out.println(o.getDireccion()+" "+Double.toString(sumaSaldo)+"\n");
}

//CONSULTA 2
System.out.println("\nConsulta 3:\nCuentas corrientes con saldo menor"
    + "que diez y la oficina\n"
    + "donde estan domiciliadas\n");
String q2 =   "select cor from H_CORRIENTE cor "
    + "where cor.saldo<10";
Query query2 = em.createQuery(q2);
List<Corriente> res2   = query2.getResultList();
for(Corriente   co2 : res2){
    System   .out.println(co2.getIBAN()+"-"+co2   .getOficina().getDireccion()+    "\n");
}


//CONSULTA 3
cabecera = "\nConsulta 4:Transferencias recibidas por la cuenta de mayor saldo\n";
System.out.println(cabecera);
Query q3 = em.createNativeQuery("SELECT * FROM H_CUENTA"
        + " where saldo in (select max(saldo) from H_Cuenta)",      Cuenta.class);
List<Cuenta>res3 = q3.getResultList   ();
for(Cuenta cc : res3){
        Query q4 = em.createNativeQuery("SELECT * FROM H_OPERACION"
        + " where cuentaOrigen_IBAN=\'"+cc.getIBAN()+"\'", Transferencia.class);
        List <Transferencia> res4 = q4 .getResultList();
        for( Transferencia t: res4){
                System.out.println("\nTransferencia con code "+t.getCodigo()+" de "
                + t.getCuentaOrigen().getIBAN()+"a"+t.getCuenta().getIBAN()+"\n");
        }
}
```

Resultados de la parte 1:

```
CONSULTAS
---------------

Consulta1:
```

Alguna oficina donde tenga domiciliadauna cuenta corriente el primer cliente alfabeticamente cuyo nombre tiene menos de 7 letras

C/San Miguel, Zgz

Consulta 2: Cantidad media movida por oficina

OFICINA Cantidad Media Movida

C/mayor, Zgz 17.5

C/San Miguel, Zgz 40.0

Consulta 3:
Cuentas corrientes con saldo menorque diez y la oficina donde estan domiciliadas

45ES987654321 C/mayor, Zgz

45ES43543454 C/San Miguel, Zgz

Consulta 4:Transferencias recibidas por la cuenta de mayor saldo

```
Hibernate: SELECT * FROM H_CUENTA where saldo in (select max(saldo) from H_Cuenta)
Hibernate: SELECT * FROM H_OPERACION where cuentaOrigen_IBAN='45ES123456789'
```

Transferencia con code 1 de 45ES123456789 a 45ES987654321

Consulta con Criteria API:

```
// Consulta 5 con Criteria API
        System.out. println("Cuentas cuyas operaciones han sido con una cantidad entre 10 y
200");
        CriteriaBuilder cb = em.getCriteriaBuilder(); //Paso 1
        CriteriaQuery< Operacion> cqry = cb.createQuery(Operacion.class); //Paso 1
        Root<Operacion> root  = cqry. from(Operacion.class); //Paso 2
        Predicate min   = cb.gt(root.< Integer>get("cantidad"),10  ); //Paso 4
        Predicate max  = cb.lt  (root.< Integer>get("cantidad"),200);
        cqry .where(cb. and(min,max));  //Paso 5

                Query qry = em.createQuery(cqry); //Paso 6
        List  <Operacion> results = qry.getResultList(); //Paso 6
        for   (Operacion o : results) {
                String IBAN = o.getCuentaOrigen().getIBAN   ();
                System.out.println(   "IBAN Cuenta Origen: "  +IBAN + " Cantidad:
"+o.getCantidad());
        }
   }
```

Resultados Criteria API:

Cuentas cuyas operaciones han sido con una cantidad entre 10 y 200

```
Hibernate:   select   operacion0_.CODIGO as   CODIGO3_, operacion0_.CANTIDAD as   CANTIDAD3_,
operacion0_. cuentaOrigen_IBAN  as   cuentaOr7_3_,   operacion0_.DESCRIPCION as   DESCRIPC4_3_,
```

```
operacion0_.FECHA as FECHA3_, operacion0_.HORA as HORA3_, operacion0_.oficina_CODIGO as oficina8_3_, operacion0_.cuenta_IBAN as cuenta9_3_, operacion0_.DTYPE as DTYPE3_ from H_OPERACION operacion0_ where operacion0_.CANTIDAD>10.0 and operacion0_.CANTIDAD<200.0
IBAN Cuenta Origen: 45ES987654321 Cantidad20.0
IBAN Cuenta Origen: 45ES43543454 Cantidad:40.0
```

5. DIFICULTADES ENCONTRADAS

Una de las principales dificultades ha sido realizar alguna consulta en Criteria API, ya que es un lenguaje de consultas muy diferente a lo que se está habituado, que es el SQL. También ha habido algún problema al hacer el código JPA con las tablas ya hechas en Oracle, ya que detectaba tablas de otros usuarios, por lo que se tenía que asignar el esquema (usuario) con el que se deseaba trabajar. También ha habido problemas a la hora de realizar consultas JPQL ya que la información por internet está dispersa y no es muy concreta.

6. ESFUERZOS INVERTIDOS

	Carlos Marañés (717788)	Nicolás Lera (721808)	Total Horas
De JPA a Oracle	6h	3h	9h
De Oracle a JPA	6h	1h	7h
Consultas	5h	10h	15h
Redacción	2h	2h	4h
TOTAL horas	19h	16h	35h

7. BIBLIOGRAFÍA

Clave primaria JPA - Último acceso 14-05-2018
https://stackoverflow.com/questions/27305950/jpa-foreign-key-that-is-also-a-primary-key-mapping

Relaciones en JPA - Último acceso 14-05-2018
www.codejava.net/frameworks/hibernate

Queries - Último acceso 14-05-2018
www.objectdb.com/java/jpa/query