Practica 1 + 2 - BASES DE DATOS 2

Hayk Kocharyan
757715@unizar.es

Pedro Tamargo Allué
758267@unizar.es

Jesús Villacampa Sagaste
755739@unizar.es

Juan José Tambo Tambo
755742@unizar.es

21 de abril de 2020

Índice

Índice de figuras

1. Esfuerzos invertidos

Los esfuerzos invertidos por cada integrante del equipo son:

- Hayk:
  - Scripts de creación de tablas y población en *Oracle*: 3 horas.
- Juanjo:
  - Instalación *DB2* y *Oracle* en máquinas virtuales: 5 horas.
  - Scripts de creación de tablas y población de base en DB4o y DB2: 7 horas.
- Jesús:
  - Traducción del diagrama *ER* a modelo relacional: 2 horas.
  - Scripts de creación de tablas de *PostgreSQL*: 3 horas.
  - Diagrama de clases *UML* para el modelo orientado a objetos: 1 hora.
- Pedro:
  - Instalación de *DB2*, *PostgreSQL* y *Oracle* en las máquinas virtuales: 3 horas.
  - Digitalización del diagrama *ER*: 1 hora.
  - Scripts de creación de las tablas en *Oracle*, en el modelo relacional y en el modelo objeto-relacional: 4 horas.
  - Memoria de la práctica: 2 horas.

2. Configuración de la máquina virtual

Para la realización de esta práctica se han utilizado las máquinas de 32 y 64 bits y su configuración ha sido igual para ambas.

Para la instalación de los *SGBD* se debía conectar por medio de ssh desde la máquina local utilizando $ssh - Xroot@dirmaquina$ para poder utilizar herramientas gráficas necesarias. Para ello, se debe configurar las interfaces de red de las máquinas de la siguiente manera:
Desde VirtualBox, añadir un adaptador "Host-only Adapter":
$Configuracion -> Network -> Adapter2 -> Host - onlyAdapter(Name : vboxnet0)$.
Para que aparezca vboxnet0:
$File -> HostNetworkAdapter -> create$
Una vez realizado lo anterior, desde la máquina virtual, se debe configurar el archivo /etc/network/interfaces y modificar la interfaz eth1 de la manera ilustrada en la Figura 1.

3. Instalación y administración básica de los SGBD

En la instalación de los *SGBD* se ha creado un usuario para la administración de la base de datos.
Para la base de datos Oracle se ha creado un usuario *oracle*. En especial en la instalación de este *SGBD* ha sido problematica en la instalación. Existen problemas con el espacio de la máquina. Al instalar todos los gestores no quedaba sitio suficiente para instalar Oracle.
Para arreglar este problema se ha aumentado el espacio del disco proporcionado con la máquina a 32GB. Pero al intentar aumentar la particion (/dev/sda1) que estaba montada en / hemos notado que existía una partición swap (/dev/sda5) que nos impedía redimensionar la primera.
Para intentar solucionar este problema se creó el nuevo espacio adicional como otra partición (/dev/sda3) y editando el fichero /etc/fstab se montó en el arranque en el directorio /oracle, directorio home del usuario oracle (necesario para la instalacion del SGBD).
Esto solucionó el problema con el espacio, pero tras proseguir con la instalación nos encontramos con un apartado de comprobación de prerrequisitos. Aquí se comprueban de manera automática que se cumplen ciertas precondiciones.

Nuestra máquina no las cumplía y, el instalador propone como solución ejecutar un script ubicado en el directorio /tmp.
Al intentarlo ejecutar la terminal nos muestra ciertos errores de formateo:

```
./orarun.sh: 186: [: true: unexpected operator
./orarun.sh: 186: [: true: unexpected operator
./orarun.sh: 848: [: unexpected operator
./orarun.sh: 864: [: unexpected operator
./orarun.sh: 882: [: unexpected operator
./orarun.sh: 903: [: true: unexpected operator
./orarun.sh: 1052: [: unexpected operator
./orarun.sh: 1057: [: unexpected operator
./orarun.sh: 1075: [: unexpected operator
./orarun.sh: 1085: [: unexpected operator
./orarun.sh: 1115: [: unexpected operator
./orarun.sh: 1143: [: unexpected operator
./orarun.sh: 1189: [: unexpected operator
./orarun.sh: 139: [: unexpected operator
./orarun.sh: 139: [: unexpected operator
./orarun.sh: 1228: [: unexpected operator
./orarun.sh: 1284: [: unexpected operator
./orarun.sh: 1342: [: unexpected operator
./orarun.sh: 1426: [: unexpected operator
./orarun.sh: 1451: [: unexpected operator
```

Al inspeccionar el archivo orarun.sh no se encuentra ningún problema. Al buscar en Internet la única explicación que se encuentra es que estamos instalando Oracle en una distro no certificada para ello. Por ello procedemos a instalar Oracle en la máquina de 32 bits.

En la instalación de *PostgreSQL* procedemos a instalar la versión *9.6*, para ello ejecutamos la orden:

```
sudo apt -y install postgresql-9.6
```

Se habrá creado el usuario *postgres*. Para la interacción con la base de datos iniciaremos sesión como este usuario. Para ello ejecutaremos:

```
sudo su - postgres
```

Para iniciar el shell interactivo de postgre utilizaremos *psqlpostgres*. Si queremos crear una base de datos ejecutaremos:

```
createdb <nombre_base_datos>
```

En el caso de que no se encuentre el binario ejecutaremos el mismo utilizando la ruta absoluta

```
/usr/local/pgsql/bin/createdb <nombre_base_datos>
```

PostgreSQL utiliza metacomandos para interactuar con su shell interactivo. El listado de metacomandos se puede encontrar aquí[1].

Para la instalación de *DB2* descargaremos los ficheros desde `https://jazz.net/downloads/DB2/releases/10.1`, utilizaremos la versión *Linux x86-64*. Una vez descargado, extraeremos el fichero con la órden:

```
tar -xzvf <filename> .
```

Una vez descomprimido accederemos al directorio y ejecutaremos la orden:

```
sudo ./db2_setup
```

Crearemos un usuario.
Añadiremos el directorio a la variable de entorno *PATH* modificando nuestro fichero */.profile*.
Para crear una nueva instancia de *DB2* ejecutaremos:

[1] `https://dataschool.com/learn-sql/meta-commands-in-psql/`

```
db2ictr <nombre_instancia>
```

Seleccionaremos la nueva instancia como la instancia por defecto utilizando:

```
set db2instance=<nombre_instancia>
```

Ahora todas las acciones se ejecutarán sobre esa instancia. Debemos cambiar de usuario al creado anteriormente.

```
su - <nombre_usuario>
```

Para crear una base de datos ejecutaremos:

```
db2 create database <database_name>
```

Iniciamos la base de datos:

```
db2start
```

Para conectarnos a la base de datos que hemos creado escribiremos:

```
db2 connect to <database_name>
```

Para crear una tabla podemos hacerlo desde fuera del shell de DB2 escribiendo:

```
db2 create table prueba(id integer not null primary key)
```

Para mostrar las tablas escribiremos:

```
db2 list tables
```

Con *db4o*, un *SGBD* orientado a objetos, no se requiere instalación ya que se ejecuta desde un fichero *.jar* proporcionado junto con el enunciado de la práctica.

4. Diseño conceptual de la base de datos

Para el diseño de la Base de Datos especificada en el enunciado se ha utilizado el siguiente diagrama entidad relación (Figura 2).

También para facilitar el diseño de la base de datos orientada a objetos utilizando *db4o* se ha creado un diagrama de clases *UML* (Figura 4).

5. Diseño lógico de una base de datos relacional

5.1. Implementación con el modelo relacional

Para la transformación del del Diagrama Entidad-Relación en un modelo relacional se ha convertido la generalización de Cuenta en dos tablas, sus subtablas *Cuenta_ahorro* y *Cuenta_corriente*, y la generalización de Transacción en sus dos subtablas Transferencia y Operación.

Las relaciones entre las entidades se han traducido en función de su cardinalidad. Las relaciones *(M:N)* se han traducido en una nueva relación.

Para la implementación en el SGBD Oracle se han utilizado tablas las tablas especificadas anteriormente y se han establecido las restricciones de integridad referencial a las tablas que hacen referencia a otras. ??

Por lo tanto, podremos convertir el Diagrama Entidad-Relación (Figura 2) en el Diagrama Relacional (Figura 3).

6. Diseño lógico de una base de datos objeto/relacional

6.1. Implementación con el modelo objeto/relacional

6.1.1. Oracle

Para la implementación del modelo *OR* con *Oracle* se han creado tantos tipos como entidades teníamos en el Diagrama Entidad-Relación, además para reflejar las relaciones bidireccionales entre dos entidades se han creado

otros tipos (listaCuentas, listaPropietarios y realizadasUdt) como tablas de referencias a otros tipos ya creados anteriormente.
Tras la creación de los tipos, procedemos a crear las tablas de los mismos. Oracle tiene como particularidad que no existen jerarquías de tablas, es decir, crearemos una tabla para el supertipo de la jerarquía y realizaremos las restricciones de integridad referencial mediante las cláusulas *FOREIGN KEY* en las tablas y *SCOPE* en las tablas anidadas.

6.1.2. PostgreSQL

Para la implementación del modelo *OR* utilizando *PostgreSQL* se ha maximizado el uso de la herencia de tablas como mecanismo de relación entre las distintas entidades planteadas en el diagrama *ER* (Figura 2).
Para la traducción de las relaciones *M:N* se han creado entidades específicas ya que, a pesar de que si se han utilizado referencias, en este gestor la propiedad del modelo OR que se ha buscado explotar es la herencia. Otros gestores como Oracle no la poseen y en el diseño referente a este gestor ya se han explotado otras características como son los tipos, tablas tipadas o arrays, por tanto no se considera primordial explotarlas también en PostgreSQL y centrar esfuerzos en observar todo lo que ofrece la herencia.
Se han observado ciertas limitaciones a la hora de la realización de pruebas, mediante inserciones y consultas que se explican en el propio documento de pruebas a través de comentarios.

6.1.3. DB2

Para la implementación con *DB2* se ha seguido un planteamiento muy similar al de *Oracle*. Se han creado tipos (*UDT*) para las distintas entidades del Diagrama Entidad-Relación (Figura 2). Tras la creación de los tipos de datos se han creado las tablas tipadas. Para poder representar las relaciones *N:M*, ha sido necesario crear tablas intermedias, que contienen una referencia a las tablas que participan en la relación.
A diferencia de *Oracle*, este *SGBD* tiene la capacidad de permitir la herencia entre tablas. De esta forma se han creado tantas tablas como tipos se han definido anteriormente.
Una diferencia con el estándar *SQL:1999* la cláusula *SCOPE* no garantiza la integridad referencial entre las tablas, por lo tanto se deben utilizar restricciones de clave ajena (*Foreign key*) para garantizarla[2].

7. Generación de datos y pruebas

Para la generación de datos y la realización de pruebas se han creado conjuntos de datos para la población y las pruebas de las distintas bases de datos con los gestores seleccionados.

Para la generación de datos, se han utilizado generadores de códigos de cuenta bancaria e IBAN. Los DNIs y los nombres han sido inventados por los integrante del equipo.

Se han probado consultas que recorrían las relaciones entre las entidades, resolviendo las referencias que aparecían entre las mismas.

8. Implementación con db4o

Siguiendo el diagrama *UML* de la Figura 4, se ha procedido a la creación de las clases *Java* correspondientes. A diferencia de la implementación en el modelo relacional de *Oracle*, no es necesario reflejar las relaciones *M:N* como una nueva entidad, en este caso se utilizarán listas (*List*).

Una de las dificultades encontradas en la implementación con *db4o* ha sido el desfase entre versiones de *Java*, por ejemplo en el uso de clases genéricas ya que la Máquina Virtual de *Java* mostraba avisos (*Warnings*) acerca de la instanciación de las clases.

[2] `https://www.ibm.com/support/knowledgecenter/SSEPGG_11.5.0/com.ibm.db2.luw.admin.structypes.doc/doc/c0006594.html`

Para la resolución de las dudas surgidas durante la implementación con este *SGBD* se ha recurrido al manual del mismo[3].

9. Comparación de los SGBD

Tras la realización de la práctica se ha comparado el funcionamiento de los distintos *SGBD* con los distintos modelos que se han estudiado.
Dado que el modelo *O-R* es un modelo nuevo para el equipo se han podido comparar la funcionalidades respecto al modelo relacional. Una de las mayores ventajas que se han observado en el modelo *O-R* es la capacidad de realizar herencias entre los tipos, debido a que reduce la complejidad de la traducción del diagrama *ER* extendido.
También, en el *SGBD* de *Oracle*, se ha observado la utilidad de que no existan jerarquías de tablas tipadas, ya que esto reduce considerablemente el número de tablas que se han de gestionar.

Con el gestor de *PostgreSQL*, se ha tomado como un aspecto negativo las limitaciones en el mecanismo de herencia. Este gestor no mantiene las restricciones de clave primaria y las de clave ajena en la herencia entre dos tablas. También durante el proceso de poblado de la base de datos la carencia de "enrutamiento automático" en las sentencias *INSERT* hizo la tarea más tediosa.

En cuanto al gestor *DB2*, un punto a favor del mismo, como en *Oracle*, es la herencia entre tipos definidos por el usuario (*UDT*), pero a diferencia de este, *DB2* permite la herencia entre tablas, un mecanismo muy útil a la hora de insertar elementos y realizar consultas. También, la facilidad de ejecutar una operación de dereferencia (*deref()* en *Oracle*), ya que el acceso a los campos en el objeto referenciado se hace mediante el operador "$->$", recordando a los punteros de lenguajes como *C* o *C++*.

Sobre el modelo *Orientado a Objetos*, se ha utlilizado el gestor *db4o*. Este *SGBD* utiliza lenguaje *Java* para almacenar los objetos. Una ventaja de este gestor es que permite almacenar los objetos de como tal, proveyendo una *API* para la persistencia de los datos en ficheros (*.db4o*).

[3] `http://www.odbms.org/wp-content/uploads/2013/11/db4o-7.10-tutorial-java.pdf`

10. Apéndice 1: Figuras

```
auto eth1
iface eth1 inet static
address 192.168.56.10
netmask 255.255.255.0
```

Figura 1: Interfaz de red para el acceso vía SSH

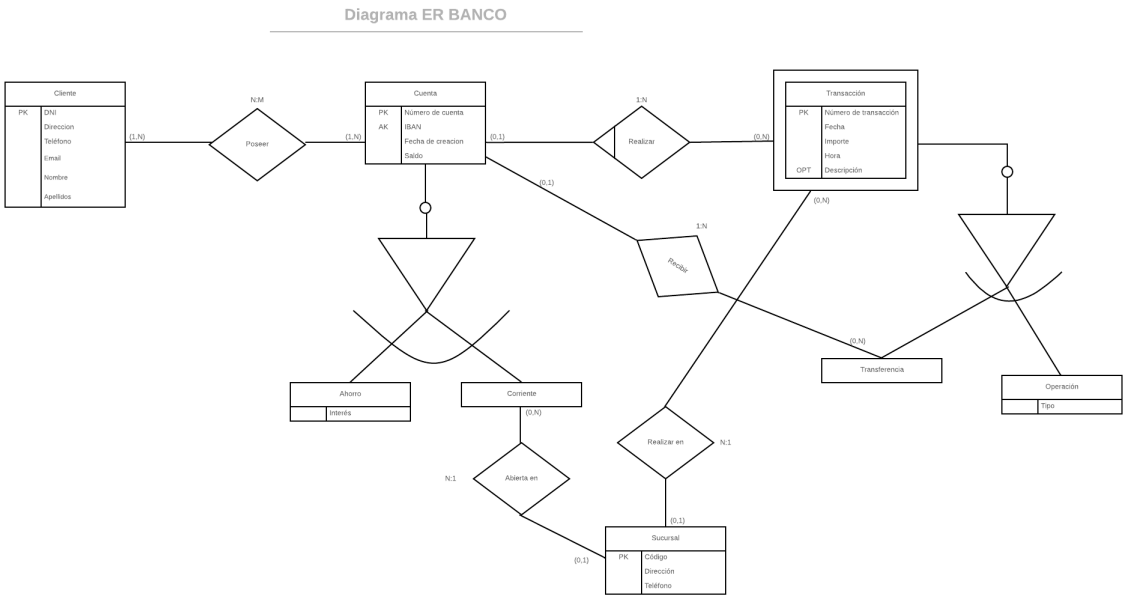

Figura 2: Diagrama E-R de la Base de Datos a diseñar

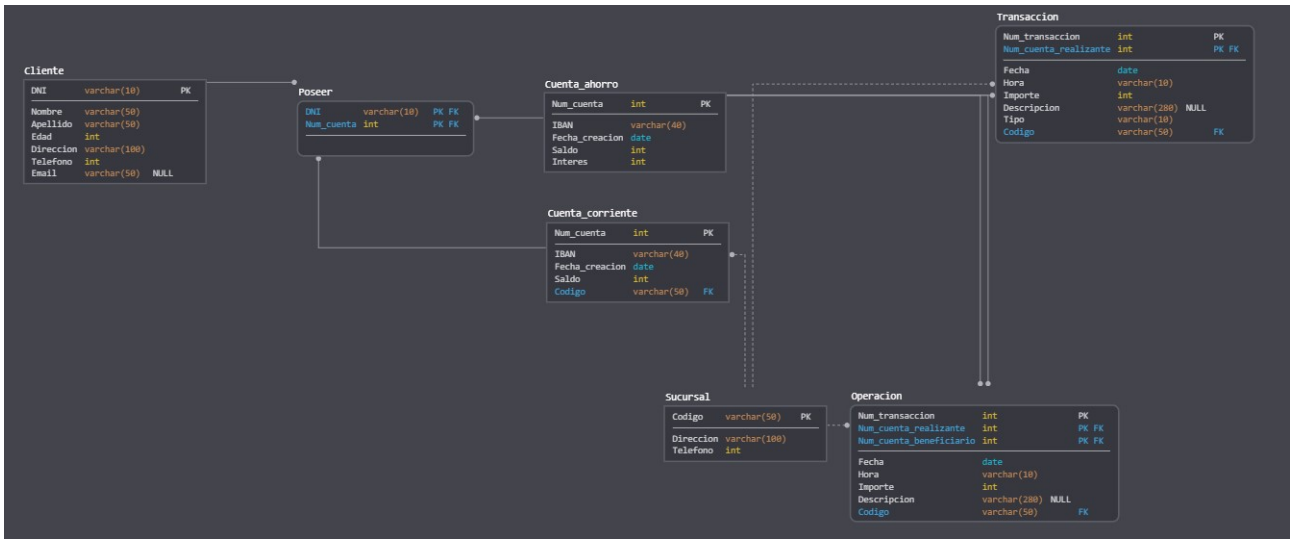

Figura 3: Diagrama del modelo relacional de la Base de Datos a diseñar

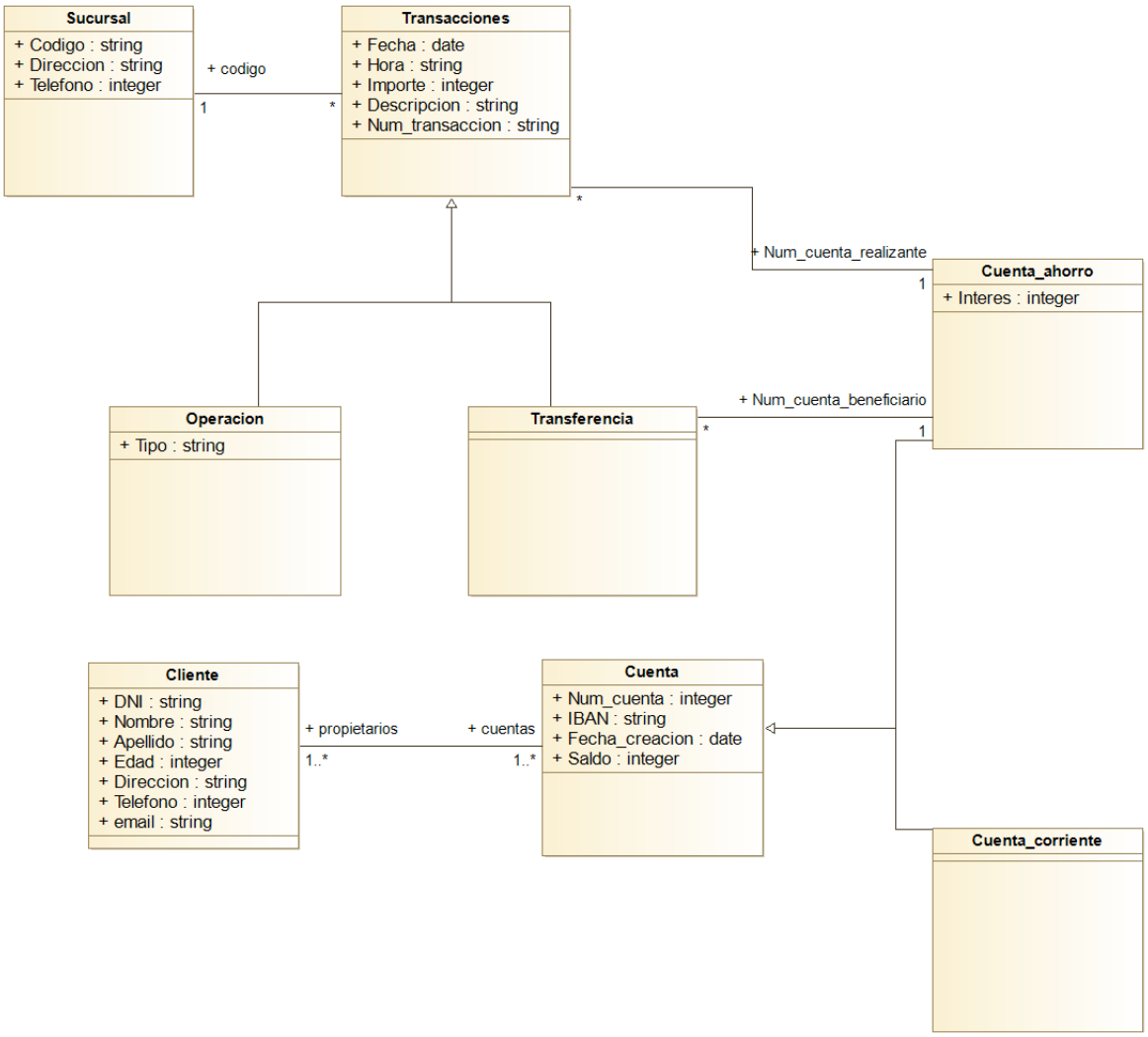

Figura 4: Diagrama de clases UML de la Base de Datos a diseñar