Índice

Objetivo de la práctica

En esta práctica el alumno utilizará la funcionalidad *dblink* de Oracle para implementar una base de datos distribuida.

Descripción del problema

Cada alumno representará una entidad bancaria. Cada entidad tendrá clientes, que pueden tener más de una cuenta. Cada cuenta almacena movimientos. Un movimiento incluye el importe, una descripción y una marca de tiempo.

Ejercicio 1 : Vistas necesarias

No son importantes las tablas utlizadas para almacenar los datos, pero deben crearse las vistas del listado 1, con los tipos de datos definidos en la tabla 1.

```
create or replace view v_cuentas(idcuenta) as
select ...;
create or replace view v_movimientos(idcuenta,marcadetiempo,euros,descripcion) as
select ...;
create or replace view v_clientes(idcliente,nombre) as
select ...;
```

Listado 1: Creación de las vistas

Campo	tipo	
idcuenta	VARCHAR(20)	El identificador de cuenta. Los 4 primeros indicarán el banco. Seguirá la cadena 0001DC. Los últimos 10 seguirán una secuencia.
idcliente	NUMBER(10)	El identificador de cliente
euros	NUMBER(15,2)	Las cantidades en euros
marcadetiempo	TIMESTAMP	Las marcas de tiempo
nombre	VARCHAR(255)	Nombre del cliente
descripcion	VARCHAR(1024)	Descripción del movimiento

Figura 1: Tipos de datos de los campos

Ejercicio 2 : Funciones internas del banco

- saldo_cuenta(idcuenta)
  El saldo de una cuenta es la suma de los importes de sus movimientos. Una cuenta sin movimientos tiene saldo 0. Si la cuenta no existe, se lanzará un error con el texto CUENTANOEXISTE
- saldo_cliente(idcliente)
  El saldo de un cliente es la suma del saldo de todas las cuentas. Un cliente sin cuentas tiene saldo 0. Si el cliente no existe, se lanzará el error CLIENTENOEXISTE

```
create or replace function SALDO_CUENTA( idcuenta IN varchar )
return number as begin
  ...
end;

create or replace function SALDO_CLIENTE( idcliente IN number )
return number as begin
  ...
  RAISE_APPLICATION_ERROR(-20002, 'CLIENTENOEXISTE');
  ...
end;
```

Listado 2: Funciones internas del banco

Ejercicio 3 : Procedimientos internos del banco

- `nuevo_movimiento(idcuenta,euros,descripcion)`
  Añade un movimiento a una cuenta, con la hora actual del sistema. Si la cuenta no existe, se lanzará el error `CUENTANOEXISTE`.
- `crear_cuenta(idcliente,idcuenta)`
  Crea una cuenta cuyo titular es el cliente especificado. Si el cliente no existe se lanzará el error `CLIENTENOEXISTE`. El identificador de cuenta será un `VARCHAR(20)`. Los 4 primeros indicarán el banco. Seguirá la cadena `0001DC`. Los últimos 10 seguirán una secuencia.
- `crear_cliente(nombre,idcliente)`
  Crea un nuevo cliente, devolviendo su identificador.

```
create or replace procedure NUEVO_MOVIMIENTO(
  idcuenta IN varchar,
  euros IN number,
  descripcion IN varchar)
as begin
  ...
  RAISE_APPLICATION_ERROR(-20002, 'CUENTANOEXISTE');
  ...
end;

create or replace procedure CREAR_CUENTA(
  idcliente IN number
  idcuenta OUT varchar)
as begin
  ...
end;

create or replace procedure CREAR_CLIENTE(
  nombre IN varchar,
  idcliente OUT  number)
as begin
  ...
end;
```

Listado 3: Procedimientos internos del banco

Ejercicio 4 : Conexión con otros bancos

1. Apunta el nombre de tu servidor, el usuario y la contraseña que usarán los demás bancos para conectarse tu base de datos en la hoja compartida
   - https://docs.google.com/spreadsheets/d/ 1Hpo3YKgWtDsmaOB1p5PX5VdGO-c3HxdzpoBsCNI7cl4/edit.
   - Recuerda que ese usuario debe tener privilegios para utilizar las vistas, funciones y procedimientos del ejercicio.
2. Crea los *dblink* necesarios para conectarte a los demás bancos.

Ejercicio 5 : Interacción entre bancos

Cada banco podrá hacer transferencias entre sus cuentas. También podrá transferir dinero desde una de sus cuentas a una cuenta de otro banco. El procedimiento `TRANSFERENCIA( idcuentaorigen, euros, descripcion, idcuentadestino)` realizará las siguientes acciones

- En la cuenta de origen se hará un movimiento negativo (que reste dinero de la cuenta), usando el método `NUEVO_MOVIMIENTO`.
- En la cuenta de destino se hará un movimiento positivo (que añada dinero a la cuenta), usando el método `NUEVO_MOVIMIENTO` del banco de destino via *dblink*, si es necesario.

- A la descripción de los momimientos se añadirá la cadena "`cuentaorigen:`*`la cuenta origen`* `cuentadestino:`*`la cuenta destino`*".
- Si la cuenta de destino no es del propio banco, se hará un cargo de una comisión en la cuenta de origen. Este cargo será de un 2 % de la transferencia, con un mínimo de 3€.
- El movimiento de la comisión tendrá como descripción "`comision por transferencia de `*`la cuenta origen`* `a `*`la cuenta destino`*"

Se tendrán en cuenta los siguientes casos de error:

- El banco de destino no existe: `BANCONOEXISTE`
- La cuenta de origen no es del propio banco: `CUENTAAJENA`
- La cuenta origen o destino no existe: `CUENTANOEXISTE`
- La cuenta origen no tiene suficiente saldo: `SALDOINSUFICIENTE`
- La transferencia es de un importe de 0€ o menor: `IMPORTEINCORRECTO`

```
create or replace procedure TRANSFERENCIA(
  idcuentaorigen IN varchar,
  euros IN number,
  descripcion IN varchar,
  idcuentadestino IN varchar )
as begin
  ...
  RAISE_APPLICATION_ERROR(-20002, 'IMPORTEINCORRECTO');
  ...
end;
```

Listado 4: Procedimientos internos del banco

Instrucciones de entrega

La autoría del trabajo es individual. Se corregirá *on-line*, ejecutando pruebas mediante conexiones de red. Los servidores Oracle deberán estar funcionando y conectados en el día que el profesor pase dichas pruebas.