

# Caso T6

1) Tenemos una base de datos relacional, que describe un modelo simplificado de un banco, con las siguientes tablas y sus correspondientes columnas:

| Tabla "clientes" |                         |   |
|------------------|-------------------------|---|
| Columna          | Tipo                    | Descripción                                 |
| cod_cliente      | integer unique not null | Código de cliente                           |
| tipo_doc         | char(1) not null        | "N"=NIF, "E"=NIE, "P"=pasaporte             |
| num_doc          | char(16) not null       | Número del documento identificador          |
| cod_pais         | char(2) not null        | Código de dos letras del país ("ES"=España) |
| apellidos        | char(40) not null       | Apellidos                                   |
| nombre           | char(40) not null       | Nombre(s) de pila                           |

| Tabla "cuentas" |                          |                        |
|-----------------|--------------------------|------------------------|
| Columna         | Tipo                     | Descripción            |
| num_cuenta      | char(20) unique not null | Número de cuenta       |
| saldo           | decimal(10,2) not null   | Saldo actual en euros  |
| interes         | decimal (10,8) not null  | Tipo de interés diario |
| sucursal        | decimal(4) not null      | Código de sucursal     |

| Tabla "movimientos" |                        |   |
|---------------------|------------------------|---|
| Columna             | Tipo                   | Descripción   |
| cuenta              | char(10) not null      | Código de cuenta  |
| fecha               | timestamp not null     | Fecha y hora de la operación                            |
| importe             | decimal(10,2) not null | Importe de la operación (puede ser positivo o negativo) |
| texto               | char(40) not null      | Texto descriptivo                                       |

| Tabla "intervinientes" |                   |   |
|------------------------|-------------------|---|
| Columna                | Tipo              | Descripción                                     |
| cliente                | integer not null  | Código del cliente interviniente                |
| cuenta                 | char(10) not null | Número de cuenta del cliente                    |
| tipo                   | char(1) not null  | Tipo interviniente: "T"=titular, "A"=autorizado |

En base al modelo de datos expuesto, escribir las sentencias SQL necesarias para realizar las siguientes operaciones:

- a) Devolver un listado con el número de todas las cuentas, acompañado del importe máximo de todos los movimientos producidos en esa cuenta, así como el importe medio de los mismos. *(Valor: 10%)*
- b) Devolver los apellidos y nombre de todos los clientes que figuren como "autorizados" para operar con alguna cuenta ordenados por apellidos y nombre. *(Valor: 10%)*
- c) Devolver un listado con el código de país y la suma total de los saldos de las cuentas en las que interviene como titular una persona de ese país, ordenado de manera que los países con más saldo figuren los primeros. *(Valor: 10%)*
- d) Devolver un listado de los números de las cuentas que han disminuido su saldo entre el timestamp \$FECHA1 y el timestamp \$FECHA2 (ambos incluidos). Las cuentas siempre tendrán un movimiento de apertura con el saldo inicial. Hay que tener en cuenta que las cuentas que no estuvieran abiertas en \$FECHA1 no deben incluirse en el listado, puesto que jamás podrán tener menos saldo que en una fecha en la cual no estaban abiertas (no se admiten saldos negativos). *(Valor: 10%)*
- e) Devolver la lista de códigos de sucursal y sus depósitos acumulados (totales de los saldos de sus cuentas) ordenada por sucursales con más depósitos a sucursales con menos. *(Valor: 10%)*

*Valor de la pregunta: 50% de la nota del caso*

2) Disponemos del siguiente código Java:

```
class MyRunnable implements Runnable {

    private boolean doStop = false;

    public synchronized void doStop() {
        this.doStop = true;
    }

    private synchronized boolean keepRunning() {
        return this.doStop == false;
    }

    public void run() {
        while(keepRunning()) {
            System.out.println("Running");

            try {
                Thread.sleep(3 * 1000); //milliseconds
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

        }
    }
}

public class MyRunnableMain {

    public static void main(String[] args) {
        MyRunnable myRunnable = new MyRunnable();

        Thread thread = new Thread(myRunnable);

        thread.start();

        try {
            Thread.sleep(10 * 1000); //milliseconds
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        myRunnable.doStop();
    }
}
```