EDAT Práctica 2

Por Erik Yuste y Lucía Martínez-Valero

Noviembre 2020

El repositorio de GitHub es: https://github.com/Erik-YO/EDAT-P2

Products Stock: Aquí simplemente encontramos los productos para los que el código fuera el mismo que el introducido por el usuario.

```
select p.quantityinstock
from products as p
where p.productcode = ?;
```

Products Find: La lógica es encontrar los productos en cuyo nombre se encuentre la cadena introducida por el usuario sin importar mayúsculas ni minúsculas (de ahí la función UPPER). Tuvimos prblemas al usar LIKE (debido a los símbolos '%') así que la solución que se nos ocurrió fue encontrar la posición en la cadena, que devuelve un numero inferior a 0 si no lo encuentra.

```
select p.productcode, p.productname
from products p
where position(UPPER(?) in UPPER(p.productname))>0
order by p.productcode;
```

Orders Open: Encontramos los orders en los que no exista una fecha de envio.

```
select o.ordernumber
from orders o
where o.shippeddate isnull
order by o.ordernumber;
```

Orders Range: Comparamos las fechas introducidas por el usuario para seleccionar las orders que están entre estas.

```
select o.ordernumber, o.orderdate, o.shippeddate
from orders o
where o.orderdate >= ? and o.orderdate <= ?
order by o.ordernumber;
```

Orders Details:

```
select o.ordernumber, o.orderdate, o.status, od.productcode, od.quantityordered,
od.priceeach, od.quantityordered*od.priceeach as subtotal
from orders o join orderdetails od on o.ordernumber=od.ordernumber
where o.ordernumber = ?
group by o.ordernumber, od.productcode, od.quantityordered, od.priceeach,
od.orderlinenumber
order by od.orderlinenumber;
```

Customers Find: Usamos la misma lógica que con la query de Products Find pero comprobando en dos campos en lugar de en uno.

```
select c.customernumber, c.customername, c.contactfirstname, c.contactlastname
from customers c
where position(UPPER(?) in UPPER(c.contactfirstname))>0
or position(UPPER(?) in UPPER(c.contactlastname))>0
order by c.customernumber;
```

Customers List Products: Hacemos joins desde la tabla customers hasta la que almacena los productos, entre ellas, la tabla orderdetails, donde se encuentra la cantidad de productos en cada pedido. Se agrupan por el codigo del producto y se suman.

```
select p.productname, sum(od.quantityordered)
from customers c
join orders o on c.customernumber=o.customernumber
join orderdetails od on o.ordernumber=od.ordernumber
join products p on od.productcode=p.productcode
where c.customernumber=?
group by p.productcode, p.productname
order by p.productcode;
```

Customers Balance: Se obtienen la suma de los pagos y la suma de los costes por separado y se restan.

```
select pay.suma-s.saldo as balance
from (select sum(pm.amount) as suma
        from payments pm
        where pm.customernumber=?
        group by pm.customernumber) as pay,
        (select sum(od.quantityordered*od.priceeach) as saldo
        from customers c join orders o on c.customernumber=o.customernumber
        join orderdetails od on o.ordernumber=od.ordernumber
        where c.customernumber=?
        group by c.customernumber) as s;
```

Paginación: Para implementarla, tras la query, almacenamos todos los resultados en un fichero temporal (para evitar tener que volver a hacer la query)