Fundamentos de Bases de Datos

Práctica 2

Miguel Lozano Alonso
Cristina Rodríguez de los Ríos Medina

Para llevar a cabo el programa solicitado lo hemos dividido en cuatro archivos ".c". En el primer archivo, *menu*, se ha incluido main y a partir de este se puede acceder al resto de menús; en los otros tres: *products*, *orders* y *customers*, se implementan cada uno de los submenús.

Implementación de Menú:

En el main se hace la conexión con la base de datos y se pasa el parámetro **dbc** como argumento a cada uno de los menús para que puedan operar sobre esta. Se imprime por pantalla el menú principal, para que el usuario pueda escoger el submenú al que quiere acceder.

Haciendo uso de un bucle *while*, el menú se ejecuta hasta que el usuario escoge la opción "exit". Al salir del bucle se realiza la desconexión de la base de datos antes de finalizar el programa.

Implementación de Products:

1. Stock

Número de unidades en stock, dado un código de producto

Query:

```
SELECT quantityinstock
FROM   products
WHERE  productcode = ?;
```

Se accede al atributo '*quantityinstock*' de la tabla '*products*' que coincida con el '*productcode*' pasado como argumento.

2. Find

Productos cuyo nombre coincide con la cadena administrada

Query:

```
SELECT productcode,
       productname
FROM   products
WHERE  productname LIKE ?
ORDER  BY productcode;
```

Obtiene el '*productcode*' y el nombre de un producto que coincida con el/ los caracteres pasados como argumento. Para ello utiliza la función *LIKE* que busca cadenas de caracteres dentro de otras.

La función LIKE identifica la cadena con el símbolo % al principio y al final de esta, por ello, en la implementación de la función en C hemos añadido ese símbolo al declarar la variable en la que se ha guardado la entrada del usuario, y posteriormente se ha colocado al final de la cadena, seguido de "\0" que indica el final de palabra.

Implementación de Orders:

1. Open

Listado de los pedidos que aún no han sido enviados

Query:

```
SELECT ordernumber
FROM   orders
WHERE  shippeddate IS NULL
ORDER  BY ordernumber;
```

Seleccionamos los pedidos en los que la '*shippeddate*' es NULL y los ordenamos de manera ascendente por su '*ordernumber*'.

2. Range

Listado de los pedidos solicitados entre las fechas introducidas por el usuario

Query:

```
SELECT ordernumber,
       orderdate,
       shippeddate
FROM   orders
WHERE  orderdate >= ?
       AND orderdate <= ?
ORDER  BY ordernumber;
```

Seleccionamos los pedidos cuyo '*orderdate*' se encuentre entre las dos fechas solicitadas.

3. Detail

Dado un identificador de pedido, se devuelven los detalles de este

Para ambas querys consultamos el '*ordernumber*' del pedido que queremos consultar. La primera query calcula el precio total del pedido. Y la segunda query muestra el '*productcode*', '*quantityordered*' y '*priceeach*' del pedido.

Query 1:

Query 2:

```
SELECT o.orderdate,
       o.status,
       Sum(o2.priceeach)
FROM    orders o
       natural JOIN
orderdetails o2
WHERE   o.ordernumber = ?
GROUP   BY o.ordernumber;
```

```
SELECT o2.productcode,
       o2.quantityordered,
       o2.priceeach
FROM    orders o
       natural JOIN orderdetails o2
WHERE   o.ordernumber = ?
ORDER   BY o2.orderlinenumber;
```

Implementación de Customers:

1. Find

Dado el nombre de contacto de un cliente se obtiene el nombre del cliente y su identificador, y el nombre y apellido del contacto

Query:

```
SELECT customernumber,
       customername,
       contactfirstname,
       contactlastname
FROM    customers
WHERE   contactfirstname LIKE ?
        OR contactlastname LIKE ?
ORDER   BY customernumber;
```

Selecciona el '*customernumber*', '*customername*', '*contactfirstname*' y el '*contactalastname*' de un cliente cuyo nombre o apellido coincida con la cadena de caracteres proporcionada como argumento,

2. List Products

Dado el identificador de un cliente se obtiene un listado de los productos solicitados por este

Query:

```
SELECT p.productcode,
       p.productname,
       Sum(od.quantityordered) total_unidades
FROM    customers c
       natural JOIN orders o
       natural JOIN orderdetails od
       natural JOIN products p
WHERE   c.customernumber = ?
GROUP   BY p.productcode
ORDER   BY p.productcode;
```

Selecciona el '*productcode*' y el '*productname*' de un producto con el total de unidades que han sido adquiridas por un cliente dado desde que se tienen registros. Para ello se ha hecho uso del operador *sum()*.

3. Balance

<u>Dado el identificador de un cliente se obtiene el saldo de este</u>

Query:

```
SELECT ( s1.pagos - s2.prod ) balance
FROM   (SELECT Sum(p.amount) pagos
        FROM   customers c
               natural JOIN payments p
        WHERE  c.customernumber = ?) AS s1,
       (SELECT Sum(od.quantityordered * od.priceeach) prod
        FROM   customers c
               natural JOIN orders o
               natural JOIN orderdetails od
        WHERE  c.customernumber = ?) AS s2;
```

Para esta consulta se han hecho dos queries. La primera, denominada s1, extrae la suma total de los pagos realizados por un cliente. La segunda, denominada s2, extrae la multiplicación de la cantidad de productos pedida con el precio de cada uno. De estas dos consultas, se realiza su diferencia, obteniendo el saldo del cliente pasado como argumento.