Íñigo Álvarez Negueruela y Daniel Cruz Navarro

PRÁCTICA 2. FUNDAMENTOS DE BASES DE DATOS

INTRODUCCIÓN:

Vamos a realizar una práctica en la que crearemos un menú en el lenguaje C, a partir del cuál puedas acceder a la base de datos para realizar distintas consultas que se nos piden.

*Hemos usado en todas las funciones el ejemplo 4 que se nos proporcionaba en las prácticas. En todas las funciones hemos creado un control en caso de que no exista una salida de la query.

PRODUCTS:

La primera función de este módulo es PrintStock. En esta función hemos creado la consulta que hay en la foto, la cuál devuelve el número de unidades de un producto en stock. Con la '?' podemos decidir que producto queremos que se busque. Para ello primero preparamos el resto de la query, luego se pide al usuario el identificador de producto que se almacena en stmt y después se ejecuta la query. Los resultados se guardan en una variable que se imprime después

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

La otra función de este módulo es PrintFind, la cuál devuelve un listado con todos los productos cuyo nombre contenga una cadena. Funciona igual que la anterior, primero preparamos la query con la '?' en el lugar del nombre del producto, luego pedimos al usuario el nombre del producto y finalmente ejecutamos la query. En esta función nos encontramos con la primera complejidad de la práctica, ya que la salida contiene dos cadenas en vez de una. Para solucionar nuestro problema usamos la función SQLBindCol para guardar el resultado en dos variables distintas que luego vamos a imprimir.

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

Íñigo Álvarez Negueruela y Daniel Cruz Navarro

ORDERS:

Primera funcion:

Seleccionamos ordernumber de todas las órdenes que aún no se han enviado y por tanto presentan el campo shippeddate como nulo, ordenamos por ordernumber.

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

Segunda:

Seleccionamos ordernumber, orderdate y shippeddate de orders, en las tuplas que estén entre dos fechas que introduce el usuario, representadas por ?, estas serán manejadas como en el ejemplo 4, aunque con alguna dificultad añadida al haber dos incógnitas.

```
SELECT ordernumber,
       orderdate,
       shippeddate
FROM   orders
WHERE  orderdate BETWEEN ? AND ?
```

Tercera:

Esta query se complica al tener que poner la suma de la cantidad total gastada, la fecha y el estatus de una orden al principio y después todos los productos de la orden. Lo que hemos hecho es juntar los 3 primeros datos con cada tupla de producto, para al final en C imprimir los 3 primeros de la tabla una vez y todas las tuplas de los demás atributos, ordenamos por orderlinenumber.

```
WITH total
     AS (SELECT Sum(ordd.priceeach * quantityordered) AS importe,
                ord.orderdate,
                ord.status
         FROM   orders ord
                NATURAL JOIN orderdetails ordd
         WHERE  ord.ordernumber = ?
         GROUP  BY ord.orderdate,
                   ord.status)
SELECT total.importe,
       total.orderdate,
       total.status,
       ord.productcode,
       ord.quantityordered,
       ord.priceeach
FROM   total,
       orderdetails ord
WHERE  ord.ordernumber = ?
ORDER  BY ord.orderlinenumber
```

CUSTOMERS:

En este módulo la primera función es Find. La consulta utilizada devuelve un listado con todos los clientes cuyo nombre o apellido de contacto contenga una cadena. Esta cadena se pide al usuario. Primero preparamos la query, después se solicita al usuario que introduzca la cadena y finalmente se ejecuta. Como en esta consulta nos encontramos con dos interrogaciones hemos usado dos veces la función SQLBindParameter para luego poder ejecutarla. Tenemos que imprimir cuatro cadenas distintas, por lo tanto la función SQLBindCol la usamos 4 veces almacenando los resultados en 4 variables que imprimimos después. En esta consulta teníamos que usar %?% lo que nos daba errores, para solucionarlo hemos incluido los paréntesis más adelante con un pequeño bucle y un char* auxiliar.

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

La segunda función es ListProducts. Se solicita el identificador de un cliente y se devuelve un listado con todos los productos solicitados por el cliente en cualquier pedido. El funcionamiento es igual que en las funciones anteriores y la única complejidad es que hay que devolver dos cadenas por lo que usamos la función SQLBindCol dos veces, almacenando los resultados en dos variables que imprimimos.

```
SELECT p.productname,
       Sum(ord.quantityordered)
FROM   products p
       JOIN orderdetails ord
         ON p.productcode = ord.productcode
       JOIN orders o
         ON ord.ordernumber = o.ordernumber
       JOIN customers c
         ON o.customernumber = c.customernumber
WHERE  c.customernumber = ?
GROUP  BY p.productname,
          p.productcode
ORDER  BY p.productcode
```

La última función es Balance, la cuál devuelve el saldo de un cliente. El funcionamiento es el mismo que en las anteriores funciones y como en nuestra consulta hemos usado dos interrogaciones hemos tenido que usar dos veces la función SQLBindParameter.

```
SELECT (SELECT Sum(pay.amount)
        FROM   payments pay
               JOIN customers c
                 ON pay.customernumber = c.customernumber
        WHERE  c.customernumber = ?) - (SELECT Sum(ord.priceeach *
                                                   ord.quantityordered)
                                        FROM   orderdetails ord
                                               JOIN orders o
                                                 ON ord.ordernumber = o.ordernumber
                                               JOIN customers c
                                                 ON o.customernumber = c.customernumber
                                        WHERE  c.customernumber = ?)
```

En las dos primeras funciones de customers hemos creado una paginación. Para implementarla hemos creado una tabla de strings, para ir guardando en cada string la salida correspondiente a como mucho 10 tuplas de la query. En caso de que haya menos de 10 no se produce la paginación y retornamos solo la primera string. En caso contrario, se produce un bucle en el que con '>' vamos hacia la siguiente página(se imprime la string con el siguiente índice), lo mismo pero hacia atrás con '<' y para salir de la paginación usamos 'e'(exit), las entradas las captamos con fgets(), la función snprintf() la utilizamos para poner cada tupla en su correspondiente string.

SPLINT

```
danicn@kelta ~/BBDD/p2  master splint -nullpass src/* inc/*
Splint 3.1.2 --- 21 Feb 2021

Finished checking --- no warnings
```

Como se aprecia, pasamos el splint sin warnings, danicn es el usuario de Daniel Cruz(para que quede claro que no es sacada de otro sitio), hemos guardado el resultado en splint.log, la fecha está incorrecta, así sale en el ordenador de Daniel.

CONCLUSIÓN

Con esta práctica hemos aprendido a acceder a las bases de datos a través de un programa escrito en C. Había muchas funciones nuevas que hemos aprendido a usar y además hemos mejorado nuestras habilidades manejando una base de datos y creando consultas.

*Los scripts funcionan en todas las funciones, pero con la paginación da errores.
*El makefile también se ha modificado, de modo que en un mismo makefile se pueda crear la base de datos y compilar nuestro menú.