FUNDAMENTOS DE BASES DE DATOS

MEMORIA DE LA PRÁCTICA 2

Fecha de Entrega: 17/11/2024

Grupo 1203

Jaime Sánchez Esteban

Cristobal Revuelta Mayordomo

Tommaso Genovese

Índice:

1. Introducción

En esta práctica se nos solicita el elaborar un programa capaz de realizar una serie de consultas predeterminadas en c con el fin de replicar una interfaz de usuario.

Para ello se nos proporcionan como material de ejemplo una serie de códigos auxiliares que demostraban como era la implementación de las funcionalidades solicitadas por separado.

En nuestra implementación, tanto el menú como las queries están basados en los ficheros menú.c y odbc-example3/4.c, los cuales hemos empleado como plantilla para el programa.

2. Material Presentado

En la entrega se pueden apreciar dos carpetas y la memoria de la práctica.

Al ser imposible ejecutar los test .sh en la versión con paginación implementada hemos decidido presentar ambas versiones.

Cualquiera de los programas puede ejecutar todas las queries solicitadas con los mismos resultados en todo momento.

La única diferencia es que, por falta de tiempo, la versión “reducida” carente de paginación no cuenta con todos los warnings del fichero splint corregidos debido a que en un principio no íbamos a entregarla al no especificarse que se entregasen las versiones por separado.

Por otra parte, las carpetas contienen todo el código y makefile correspondientes para su correcta compilación y ejecución. Solo la versión sin paginación incluye los test empleados de ejemplo, siendo estos tanto los archivos correspondientes como los comandos relacionados del fichero makefile del programa.

3. Ficheros Makefile

Como se comentó previamente, ambas versiones de la práctica contienen un fichero makefile diferente, los cuales comparten los siguientes comandos:

all: Elimina y crea una base de datos classicmodels, requiere el fichero classicmodels.sql en el mismo directorio para ejecutarse correctamente.

createdb: Crea la base de datos

dropdb: Elimina la base de datos

dump: Crea un dump de la base

destore: restaura la base de datos

shell: Abre la shell de comandos

compile: Elimina todos los archivos.o y el ejecutable y los compila nuevamente

clean: Elimina los archivos.o y el ejecutable

splint: Ejecuta el comando splint y luego dumpea el retorno en un fichero splint.log

Adicionalmente, la versión sin paginación contiene los siguientes comandos auxiliares para los test proporcionados:

test_all: Ejecuta todos los test dados de forma consecutiva

test_products: Ejecuta todos los tests relacionados con el submenu products

test_products: Ejecuta todos los tests relacionados con el submenu orders

test_customers: Ejecuta todos los tests relacionados con el submenu customers

give-permisions: Le da permisos de ejecucion a todos los scripts.sh del directorio

4. Estructura de la práctica:

El código entregado se puede dividir en dos secciones independientes, los menús y las consultas. En este apartado cubriremos los detalles de la implementación de ambos, si desea mirar la lógica detrás de las consultas SQL implementadas esta se encontrará en el apartado 5.

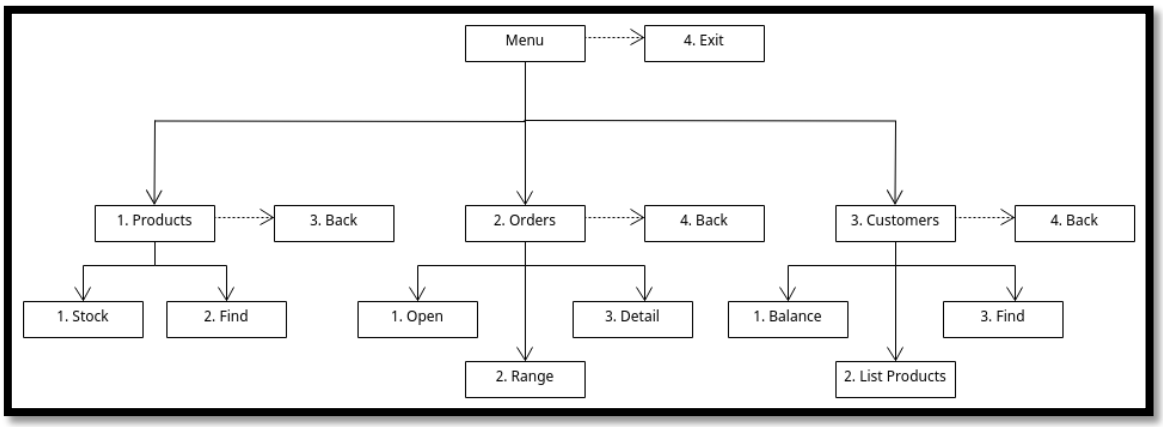
4.1 Menús

Los menús se encuentran divididos en un menú principal y tras submenús, permitiendo el tránsito entre ellos.

El menú principal permite acceder a los submenús, dependiendo del número que sea introducido, se llevará al usuario al submenú de products, orders o customers. Una vez dentro de estos submenús, se podrán ejecutar las diferentes queries mediante llamadas a funciones definidas en el fichero propio de cada submenú.

Para escoger la consulta deseada, el usuario deberá dentro del submenú introducir nuevamente un número correspondiente con las opciones de queries mostradas.

A continuación, adjuntamos un diagrama que incluye los potenciales flujos por los menús:



4.2 Implementación de las Consultas

A continuación, describiremos el funcionamiento general de la implementación de consultas implementadas en la versión no paginada de la práctica.

En primer lugar, todas las funciones de las queries siguen una misma estructura, que se modifica en ciertos puntos concretos para adaptarse a los tipos de dato, o algún requerimiento de cada queries en cuestión.

Durante esta explicación emplearemos como ejemplo la query Stock de products:

Lo primero que hacemos es inicializar las variables que serán necesarias para que la implementación de la query se lleve a cabo de forma correcta.

```
    SQLHENV env = NULL;
    SQLHDBC dbc = NULL;
    SQLHSTMT stmt = NULL;
    int ret;        /* odbc.c */
    SQLRETURN ret2; /* ODBC API return status */
    char productcode[BUFFER_LENGHT] = "\0";
    int quantityinstock = 0;
```

Tras esto, conectamos con la base de datos a través de llamada a función y comprobamos que la conexión se ha llevado a cabo con éxito.

```
    ret = odbc_connect(&env, &dbc);
    if (!SQL_SUCCEEDED(ret))
    {
        return EXIT_FAILURE;
    }
```

Una vez conectado a la base de datos, preparamos la consulta que queramos llevar a cabo. Para ello creamos un handle que nos permitirá gestionar los resultados de las consultas y lo asignamos al puntero stmt. Escribimos la consulta que deseamos que se ejecute. Vemos que aparece una interrogación que será sustituida por el valor que introduzca el usuario, en este caso el productcode del producto que queramos. En última instancia, comprobamos que la preparación de la consulta se haya realizado con éxito.

```
ret = SQLAllocHandle(SQL_HANDLE_STMT, dbc, &stmt);
     ret = SQLPrepare(stmt, (SQLCHAR *)"SELECT quantityinstock FROM products WHERE productcode
= ?;", SQL_NTS);
    if (!SQL_SUCCEEDED(ret))
    {
        odbc_extract_error("", stmt, SQL_HANDLE_ENV);
        return ret;
    }
```

Una vez preparada, ejecutamos la consulta pidiendo al usuario el valor que asignaremos a la interrogación y leyéndolo para asignarlo al lugar en el que corresponda. Tras esto ejecutamos la query, haciendo control de errores y señalando el tipo de error en caso de haberlos. Si todo se ejecuta adcuadamente, se imprime el resultado de la query.

```
    printf("Enter productcode > ");
    (void)fflush(stdout);
    scanf("%s", productcode);
    (void)SQLBindParameter(stmt, 1, SQL_PARAM_INPUT, SQL_C_CHAR, SQL_C_CHAR, 0, 0, productcode,
0, NULL);
    if (SQLExecute(stmt) != SQL_SUCCESS)
    {
        printf("ERROR EXECUTING QUERY\n");
    }
    if (SQLBindCol(stmt, 1, SQL_C_SSHORT, &quantityinstock, BUFFER_LENGHT, NULL) != SQL_SUCCESS)
    {
        printf("ERROR WHILE BINDING COLUMNS\n");
    }
    printf("Fetching results from query...\n");
    /* Loop through the rows in the result-set */
    while (SQL_SUCCEEDED(ret = SQLFetch(stmt)))
    {
        printf("quantityinstock = %d\n", quantityinstock);
    }
```

Tras la ejecución de la consulta, cerramos el cursor y comprobamos que se haya hecho adecuadamente para que stmt pueda volver a ejecutar una nueva consulta y evitar posibles bloqueos de la base de datos que se puedan producir porque el cursor esté en alguna fila o columna en concreto.

```
    ret2 = SQLCloseCursor(stmt);

    if (!SQL_SUCCEEDED(ret2))
    {
        odbc_extract_error("", stmt, SQL_HANDLE_STMT);
        return ret;
    }
    printf("\n");
```

En caso de no querer hacer más consultas es necesario liberar la memoria que habíamos asignado para el handle de las querys y comprobar que esto se lleve a cabo con éxito.

```
    ret2 = SQLFreeHandle(SQL_HANDLE_STMT, stmt);
    if (!SQL_SUCCEEDED(ret2))
    {
        odbc_extract_error("", stmt, SQL_HANDLE_STMT);
        return ret;
    }
```

Por último, una vez hecho todo lo necesario para ejecutar nuestra query, nos desconectamos de la base de datos y devolvemos el valor de éxito para indicar al programa que todo ha ido como debía y la query se ha ejecutado sin complicaciones.

```
ret = odbc_disconnect(env, dbc);
if (!SQL_SUCCEEDED(ret))
{
    return EXIT_FAILURE;
}

return EXIT_SUCCESS;
```

4.3 Implementación de las Consultas Paginadas

La lógica de la versión con paginación es bastante similar, en las queries en las que tenía sentido implementarla (aquellas con más de un retorno).

Para ello hemos alterado las queries, añadiendo al final de todas LIMIT y OFFSET, empleando el primero para solo permitir diez resultados en pantalla y el segundo para navegar por ellos mediante nuevas consultas.

Para implementar esto hemos desplazado a una sección anterior la entrada del input del usuario, intercambiándolo con la preparación de la query y hemos introducido esta última, junto con los Binds de parámetros, el execute, el retorno de la query y el close cursor dentro de un bucle infinito.

En el mismo bucle, después de la impresión de resultados, hemos añadido la siguiente sección:

```
    printf("\nPress '+' to go to the next page, '-' to go to the previous page, or 'q' to quit.\n");
    user_input = getchar();

    if (user_input == '+')
    {
        current_offset += page_size;
    }
    else if (user_input == '-')
    {
        if (current_offset > 0)
        {
            current_offset -= page_size;
        }
        else
        {
            printf("You are already at the first page.\n");
        }
    }
    else if (user_input == 'q')
    {
        break;
    }
    else
    {
        printf("Invalid input. Please try again.\n");
    }
```

Donde damos instrucciones de como navegar al usuario y en función de lo introducido aumentamos o disminuimos el offset antes de reiterar el bucle, haciendo una nueva consulta con este valor ajustado y así obteniendo una página nueva.

5. Queries

A continuación, explicaremos las queries implementadas desde un punto de vista de SQL.

Las consultas a continuación no contarán con el LIMIT y OFFSET de la versión paginada, puesto que no son relevantes para su funcionamiento básico al solo ser usadas como herramientas para lograr la paginación.

5.1 Products Stock

La query implementada es la siguiente:

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

En ella simplemente estamos buscando el valor de quantityinstock de products que corresponde a un determinado productcode.

5.2 Products Find

La query implementada es la siguiente:

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

Esta query obtiene dos valores, el productname y el productcode de un producto cuyo nombre contenga un determinado carácter o cadena de caracteres.

Normalmente la expresión LIKE requiere el uso de “%” para poder hacer lo solicitado, pero debido a un problema con la función empleada para preparar las queries nos vemos forzados a concatenar dichos asteriscos con la cadena introducida por el usuario de la siguiente forma:

```
    printf("Enter productname > ");
    (void)fflush(stdout);
    (void)scanf("%s", producttosearch);
    char searchTerm[BUFFER_LENGHT];
#pragma GCC diagnostic push
#pragma GCC diagnostic ignored "-Wformat-truncation"
    (void)snprintf(searchTerm, sizeof(searchTerm), "%%%s%%", producttosearch);
#pragma GCC diagnostic pop
```

5.3 Orders Open

La query implementada es la siguiente:

```
SELECT o.ordernumber
FROM orders o
WHERE o.shippeddate IS null
ORDER BY o.ordernumber;
```

En ella simplemente obtenemos los ordernumber de las ordenes que no han sido enviadas todavía, distinguiendo esto mediante el requisito de que su shippeddate sea nulo. Posteriormente, los resultados se ordenan en función de su número de orden.

5.4 Orders Range

La implementación de esta query es la siguiente:

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

En ella obtenemos el numero de orden y la fecha de pedido y envió de todas las ordenes que se encuentren dentro de un intervalo de fechas.

5.5 Orders Detail

Para implementar esta consulta hemos requerido de las siguientes queries:

La primera simplemente obtiene la fecha de pedido y el estado de este de una determinada orden.

```
SELECT o.orderdate, o.status
FROM orders o
WHERE o.ordernumber = ?;
```

La segunda obtiene el precio total de la orden solicitada multiplicando el precio de cada producto por la cantidad pedida.

```
SELECT SUM(od.priceeach * od.quantityordered) AS total_amount
FROM orderdetails od
WHERE od.ordernumber = ?;
```

La tercera y última obtiene el productcode, cantidad solicitada y precio de cada unidad de cada producto.

```
SELECT od.productcode, od.quantityordered, od.priceeach
FROM orderdetails od
WHERE od.ordernumber = ?
ORDER BY od.orderlinenumber;
```

La consulta ejecutada por estas queries potencialmente podría haber sido solucionada por una única solicitando todo lo que se nos pide en una sola y con c solo imprimiendo lo que se nos pide en cada fila, pero nos decantamos por realizarlo por separado e imprimir el resultado individual de cada consulta por cada fila.

5.6 Customers Find

Esta query ha sido implementada de la siguiente forma:

```
SELECT c.customernumber, c.customername, c.contactfirstname, c.contactlastname
FROM customers c
WHERE (c.contactfirstname LIKE ?) OR (c.contactlastname LIKE ?);
```

De forma similar a Find de products, obtenemos el nombre, nombre de contacto y apellido de un cliente junto con su número correspondiente en función de si su nombre de contacto o apellido contienen una determinada cadena.

5.7 Customers List Products

Esta query ha sido implementada de la siguiente forma:

```
SELECT p.productname, SUM(od.quantityordered) AS total_amount
FROM orderdetails od JOIN products p ON p.productcode = od.productcode
JOIN orders o ON od.ordernumber = o.ordernumber
WHERE o.customernumber = ?
GROUP BY p.productcode
ORDER BY p.productcode;
```

Esta query obtiene el nombre de producto y la cantidad solicitada por un determinado cliente, ordenando el resultado obtenido por el código de los productos mostrados.

5.8 Customers Balance

Finalmente, esta query ha sido implementada de la siguiente forma:

```
SELECT
    (SELECT SUM(p.amount)
    FROM payments p
    WHERE p.customernumber = ?)
     -
    (SELECT SUM(od.priceeach * od.quantityordered)
    FROM orders o JOIN orderdetails od ON o.ordernumber = od.ordernumber
    WHERE o.customernumber = ?) AS balance;
```

En ella devolvemos el saldo de un cliente marcado por su customernumber y calculado mediante la resta del total de sus pagos menos el total de sus productos comprados.