ESTRUCTURAS DE DATOS: PRÁCTICA 3

Arquitectura de un sistema de gestión de bases de datos: Esecuele

Autores: Emilio Cuesta y Adrián Fernández. Grupo 1201

Tareas a realizar:

1. Añadir los tipos de dato double y long:

```
typedef enum {
    INT, /*integer*/
    STR, /*string*/
    DBL, /*double*/
    LNG, /*long*/
} type_t;
```

Modificación en la estructura type_t

Para que las funciones de type.c utilizaran los nuevos tipos solo hubo que ampliar el switch para que incluyera los casos en los que el tipo era double o long. En ese caso las funciones actuaban prácticamente igual que con enteros.

2. Implementar table y record:

```
struct table_ {
  FILE* file;
  int ncols;
  type_t* types;
  long first_pos;
  long last_pos;
};
```

Estructura de table

La estructura table contiene un puntero al fichero que contiene los registros, el número de columnas de la tabla, un array con los tipos de elementos de cada columna y dos long, uno que indica la posición del fichero donde empiezan los registros y otro que indica donde terminan.

Las funciones de table se encargan de gestionar los archivos donde están guardadas las tablas y sus registros. Las únicas funciones que trabajan con registros son table_insert_record y table_read_record, el resto se ocupan sólo de las cabeceras de los ficheros (compuestas por ncols y types).

```
struct record_ {
    int ncols;
    void** values;
    long next;
};
```

Estructura de record

Esta estructura contiene una fila de valores de una tabla junto a su número de columnas y a un long que indica la posición del siguiente registro en el fichero.

Las funciones de record son muy básicas y no requieren de explicación ninguna.

Programa de prueba:
Se ha realizado un programa para comprobar que la implementación de los módulos table y record es correcta. El código del mismo está presente en la carpeta development bajo el nombre de “prueba.c”. Se ha modificado el Makefile para que genere el archivo ejecutable en la carpeta install al igual que hace con esecuele.c. El resultado de este programa es el siguiente:

```
emi@EmilioPC:~/edat/practica3/esecuele/install$ ./prueba
table_create OK
table_open OK
table_ncols OK
table_types OK
table_first_pos OK y table_last_pos OK
table_insert_record OK y table_read_record OK
record_next OK
Funciones de liberacion de memoria OK
Todo funciona correctamente
```

3. Implementar COUNT, UNION, LIMIT y OFFSET:

En cada operación se adjunta una captura de pantalla de una ejecución sobre la base de datos bank_db.

Count:
La operación count contiene una suboperación y un entero donde se guarda el número de registros de dicha suboperación. En operation_count_create se crea memoria para una sola columna de tipo entero que se inicializa a -1. En operation_count_next cuenta el número de llamadas a next que se pueden realizar si la cuenta no ha sido iniciada anteriormente.

```
emi@EmilioPC:~/edat/practica3/esecuele/install$ ./esecuele query bank_db
q> accounts SEQUENTIAL COUNT
15
(1 rows retrieved)
```

Union:
La operación union contiene dos suboperaciones y un flag que indica en cual de ambas operaciones se encuentra. En operation_union_create se reserva memoria para tantas columnas como tenga la primera suboperación siempre y cuando las dos suboperaciones tengan el mismo número de columnas. La función operation_union_next hace next en la primera suboperación hasta que que devuelve 0, en ese momento se actualiza el flag para indicar que se ha pasado a la segunda suboperación y en la próxima llamada a operation_union_next se hará next en la segunda suboperación. La función operation_union_get hace get de la suboperación uno o dos dependiendo del valor del flag. Es importante mencionar que la operación UNION solo se puede realizar sobre dos operaciones cuyo resultado tiene el mismo número de columnas y que son del mismo tipo, es decir, que tienen una estructura similar.

```
emi@EmilioPC:~/edat/practica3/esecuele/install$ ./esecuele query bank_db
q> clients SEQUENTIAL clients SEQUENTIAL UNION
1       Smith
2       Jones
3       Simpson
4       Cooper
5       Johnson
6       Macdonald
7       White
8       Lee
9       Brandon
10      Simmons
1       Smith
2       Jones
3       Simpson
4       Cooper
5       Johnson
6       Macdonald
7       White
8       Lee
9       Brandon
10      Simmons
(20 rows retrieved)
```

Limit:

La operación limit contiene una suboperación, un entero límite y un entero que indica la posición. La función create reserva para esta operación la misma memoria que para la suboperación. En next solo hace next de la suboperación si su posición (la cual incrementa en uno cada vez que se hace un next) es inferior al límite.

```
emi@EmilioPC:~/edat/practica3/esecuele/install$ ./esecuele query bank_db
q> accounts SEQUENTIAL 3 LIMIT
1       Brooklyn        456
2       Queens  1422
3       Brooklyn        2411
(3 rows retrieved)
```

Offset:

La operación offset contiene una suboperación, un entero offset y un entero que indica la posición. Funciona de manera idéntica a limit, salvo en que el next se salta los valores anteriores al offset en la primera llamada.

```
emi@EmilioPC:~/edat/practica3/esecuele/install$ ./esecuele query bank_db
q> accounts SEQUENTIAL 3 OFFSET
4       Queens  245
5       Queens  778
6       Bronx   123
7       Bronx   1000
8       Manhattan       555
9       Manhattan       1877
10      Brooklyn        6545
11      Manhattan       888
12      Manhattan       1211
13      Queens  12
14      Brooklyn        741
15      Queens  471
(12 rows retrieved)
```

4. Crear libros_db mediante libros.bash y queries:

Para crear la base de datos “libros_db”, se utilizarán en primer lugar los comandos de creación de tablas de esecuele. Posteriormente, se procederá a poblar la base de datos utilizando los ficheros de la librería que se proporcionaron para realizar la práctica 2. Todos estos ficheros se encuentran en esecuele/development/files. Los comandos utilizados se muestran en las dos próximas imágenes.

```
$COMMAND define libros_db << EOF
TABLE isbns_precios 2 STR DBL
TABLE libros 8 STR STR STR STR STR STR STR STR
TABLE usuarios 6 INT STR STR STR LNG STR
TABLE ventas 4 INT LNG STR STR
EOF

$COMMAND insert libros_db << EOF
COPY isbns_precios ./files/isbns_precios.txt
COPY libros ./files/LIBROS_FINAL.txt
COPY usuarios ./files/usuarios.txt
COPY ventas ./files/ventas.txt
EOF
```

Modificación del código de bank.bash para crear libros_db

```
esecuele:   $(MAINOBJS)
    $(LNK) -o ../install/esecuele $(MAINOBJS)
    cp bank.bash ../install
    cp -r example_files ../install/
    cp libros.bash ../install
    cp -r files ../install/


wvs:    $(ALTOBJS)
    $(LNK) -o ../install/esecuele $(ALTOBJS)
    cp bank.bash ../install
    cp -r example_files ../install/
    cp libros.bash ../install
    cp -r files ../install/
```

Modificación del makefile para que genere libros.bash

Ejecutando el archivo libros.bash (dándole permiso como ejecutable o mediante el comando “bash libros.bash” se realizará todo este proceso automáticamente. Además se ejecutarán las dos consultas propuestas. Para salir terminar con la ejecución, es recomendable pulsar **ctrl+D.**

Queries en **SQL**:
En el enunciado se proponen dos consultas para comprobar que ESECUELE puede funcionar de forma equivalente a SQL. Son las siguientes:

1. Lista de libros comprados por "jack".

La consulta en SQL sería:

```
SELECT l.titulo
FROM  libro as L, venta as V, usuario as U, rincluido as R
WHERE l.isbn = R.isbn AND R.idventa=V.idventa AND V.idusuario=U.idusuario AND U.scrname='jack'
```

Vemos que funciona utilizando pgAdmin. Los resultados obtenidos son ligeramente distintos porque, a la hora de introducir los datos de los ficheros en la base de datos para la práctica 2, fue necesario modificar o eliminar basantes tuplas.

	titulo character varying(200)
1	What We Become
2	The Fencing Master
3	The Flanders Panel
4	Manufacturing Consent: The Political Economy of the Mass Media
5	Manufacturing Consent: The Political Economy of the Mass Media
6	Superpowers in Collision: The Cold War Now
7	The Iron Wall: Israel and the Arab World
8	Lucky Jim (Penguin Modern Classics)

2. Números de libros comprados por 'jack'

```
SELECT COUNT(l.titulo)
FROM  libro as L, venta as V, usuario as U, rincluido as R
WHERE l.isbn = R.isbn AND R.idventa=V.idventa AND V.idusuario=U.idusuario AND U.scrname='jack'
```

Y el resultado, aunque distinto al que obtendremos utilizando esecuele por lo explicado anteriormente, es el siguiente:

	count bigint
1	288

Queries traducidas a **esecuele**:

1. Lista de libros comprados por "jack".
usuarios SEQUENTIAL 1 STR jack C_COLEQCTE SELECT INT 0 P_COL STR 1 P_COL 2 PROJECT ventas SEQUENTIAL PRODUCT 0 3 C_COLEQCOL SELECT STR 1 P_COL STR 4 P_COL 2 PROJECT libros SEQUENTIAL PRODUCT 1 9 C_COLEQCOL SELECT STR 0 P_COL STR 3 P_COL 2 PROJECT

2. Números de libros comprados por 'jack'
usuarios SEQUENTIAL 1 STR jack C_COLEQCTE SELECT INT 0 P_COL STR 1 P_COL 2 PROJECT ventas SEQUENTIAL PRODUCT 0 3 C_COLEQCOL SELECT STR 1 P_COL STR 4 P_COL 2 PROJECT libros SEQUENTIAL PRODUCT 1 9 C_COLEQCOL SELECT STR 0 P_COL STR 3 P_COL 2 PROJECT STR 1 P_COL 1 PROJECT COUNT

Las queries han sido introducidas en libros.bash para que se realicen de forma automática cada vez que se ejecuta el programa. Estos son los ejemplos de ejecución:

QUERY 1:

Tan solo se adjunta una parte de la salida, pues esta da como resultado 361 tuplas. Se incluye la primera columna con el nombre del comprador para poder comprobar que todos los libros mostrados por pantalla han sido comprados por jack.

```
jack    The Fencing Master
jack    Some Account or the Life and Writings: Of Lope Felix De Vega Carpio (Classic Reprint)
jack    El Puente De Los Asesinos
jack    El asedio = The Siege
jack    Don Quijote de la Mancha I / Don Quixote de la Mancha (Biblioteca Didactica Anaya)
jack    Powers and Prospects: Reflections on Nature and the Social Order
jack    Capitaine Alatriste T.6 - Corsaires du Levant
jack    Un Dia de Colera
jack    Esperanzas y Realidades
jack    Three Exemplary Novels/Tres Novelas Ejemplares: A Dual-Language Book (Dover Dual Language Espa◆ol)
jack    Cervantes' "Don Quixote" (The Open Yale Courses)
jack    Selected Commercial Statutes, For Sales and Contracts Courses: 2 (Selected Statutes)
jack    Selected Commercial Statutes, For Sales and Contracts Courses: 2 (Selected Statutes)
jack    Don Quijote: A new translation by Burton Raffel
jack    Don Quijote: A new translation by Burton Raffel
jack    The Sound Pattern of Ingl◆s
jack    Inside Syria: The Backstory of Their Civil War and What the World Can Expect
jack    Inside Syria: The Backstory of Their Civil War and What the World Can Expect
jack    Don Quixote: [Complete & Illustrated]
jack    Don Quixote: [Complete & Illustrated]
jack    Fama Postuma: La Discreta Enamorada, Comedia En Tres Actos...
jack    La guerra civil contada a los j◆venes
jack    Patente de Corso
jack    Poes◆a selecta / Selected Poetry (Letras Hisp◆nicas / Hispanic Literature)
jack    Dona Berta (Cervantes & Co. Espa◆olClassics)
jack    Dona Berta (Cervantes & Co. Espa◆olClassics)
jack    La Mayor Vitoria
jack    Donde se posa el resplandor del sol
jack    Le Gentilhomme Au Pourpoint Jaune
jack    El sol de breda (alatriste iii)
jack    Zycie jak w Madrycie
jack    El Villano en su Rincon
(361 rows retrieved)
```

QUERY 2:

Como vemos, el resultado que muestra la imagen concuerda con el valor de la query anterior, pues la operación COUNT cuenta el número de tuplas que da como resultado una consulta.

```
q> 361
(1 rows retrieved)
```

Como SQL es reconocido dentro del mundo de las Bases de Datos y ha sido actualizado y renovado continuamente durante muchos años, las consultas en este lenguaje resultan mucho más sencillas sintáctica y visualmente hablando que las realizadas en esecuele. Es algo lógico, pero atendiendo a los resultados vemos como esecuele es perfectamente válido, pese a sus limitaciones, si se utilizan las operaciones implementadas de forma adecuada.