Fundamentos de Bases de Datos

Práctica 3

David Brenchley Uriol

Javier San Andrés de Pedro

Introducción

La realización de esta práctica ha sido enfocada a entender la gestión a nivel interno de una base de datos. Aunque la implementación llevada a cabo es una expresión mínima de una base de datos utilizada a nivel profesional, nos ha transmitido una idea inicial de cómo se pueden implementar.

En concreto, el procedimiento implementado se basa en dos archivos, uno con los datos propiamente dicho, y otro un índice simple basado en un árbol binario, aunque no completo. El mantenimiento y gestión de los datos borrados sigue una estructura de lista enlazada, tanto el archivo de datos como en el índice. En el caso de los datos existe un puntero inicial que apunta al primer registro borrado, que contendrá el puntero al siguiente registro borrado y así sucesivamente. En cuanto al índice, debido a la longitud fija de dichos registros, en vez de un puntero de memoria se utiliza un entero (ID) que indique la posición respecto a los demás registros. No se debe confundir el book_id, que identifica un libro, con el node ID que identifica la posición del nodo en el fichero del índice.

Además de la propia implementación, también hemos ampliado nuestros conocimientos sobre funciones de escritura y lectura en ficheros a nivel de byte. La política de escritura en disco duro se basa en la modificación inmediata de los archivos por cada operación, en vez de cargar en memoria los cambios realizados. Esto permite una lectura y escritura mucho más coordinada por varios programas que accedan a la base de datos.

Igual que en la práctica anterior, se ha desarrollado un menú que permite al usuario acceder a una base de datos y añadir información de manera sencilla y cómodo, sin necesidad de conocer el funcionamiento detrás de la BD.

Finalmente, se ha profundizado en el uso de la metodología de TDD (test-driven development), desarrollando el código de manera secuencial y testeando cada función según el orden de creación. El beneficio de dicho método se palpita en el desarrollo de funciones que se llamen entre sí, evitando una comprobación de todo el código por cada error y limitándolo a la función inicial en la que se origine.

Funciones desarrolladas

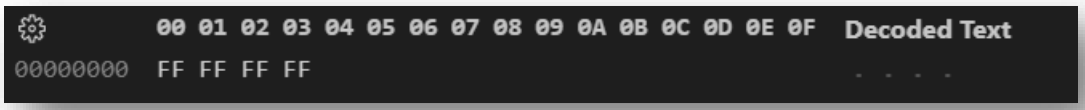
Para a implementación individual de cada función, se ha procedido al desarrollo y posterior testeo de cada una de manera individual. A continuación se explica cada una de ellas:

```
void replaceExtensionByIdx(const char *fileName, char *indexName)
```

Dicha función se encarga de copiar el nombre de "fileName" a "indexName" sustituyendo la terminación ".dat" de "fileName" por".idx". Para ello nos hemos apoyado en la librería string.h y las funciones strcpy y strcat.

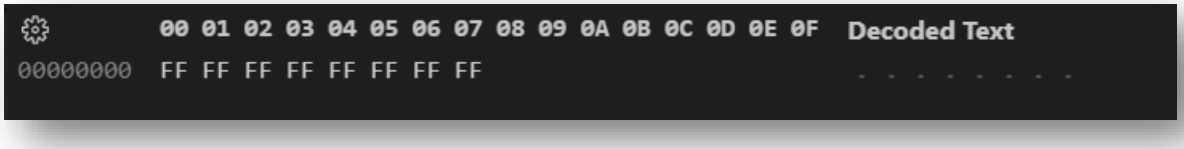
```
bool createTable(const char *tableName)
```

Dicha función se encarga de crear una nueva tabla de datos con el nombre "tableName". Para ello, primero comprueba si existe el fichero, en cuyo caso devuelve "true" directamente, pues considera que además del correspondiente fichero de datos también existe su correspondiente índice. Sin embargo, en el caso que no exista, crea un nuevo fichero e inicializa el puntero a los registros borrados a -1, para posteriormente crear el índice con el ID de la raíz y primer registro borrado también a -1. En caso de error devuelve "false", sino devuelve "true".




```
bool createIndex(const char *indexName)
```

Función que se encarga de crear el índice, utiliza la misma metodología que createTable, desde la cual se llama. En caso de error devuelve "false", sino devuelve "true".



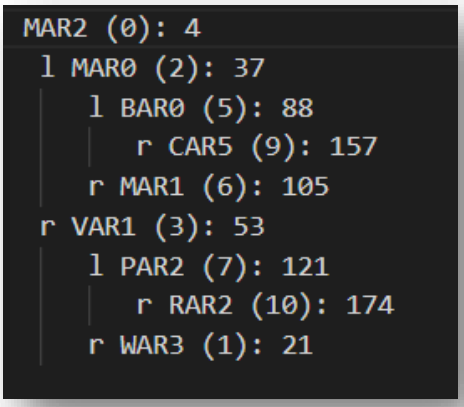
```
void printnode(size_t _level, size_t level, FILE *indexFileHandler, int node_id,
char side)
```

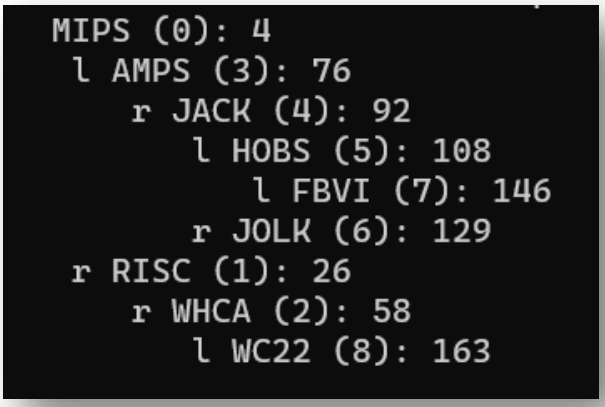
Printnode, como bien indica su nombre, imprime la información del nodo con el ID correspondiente y que pertenezca a indexFileHandler. Además imprime el lado (respecto a su nodo padre) en el que se localiza.

```
void printTree(size_t level, const char *indexName)
```

PrintTree imprime la estructura de árbol binario que representa el índice (el árbol lo recorre en preorder). Además también tiene un argumento "level" que indica el numero de niveles del árbol que se quieren imprimir.

A continuación tenemos dos ejemplos de dicha impresión (l=left, r=right)






```
bool findKey(const char *book_id, const char *indexName, int
*nodeIDOrDataOffset)
```

Findkey realiza una busqueda binaria en el índice basándose en book_id como calve de comparación. En el caso de encontrar la clave en el índice, guarda en nodeIDorDataOffset el offset del libro en el archivo de datos y devuelve "true". En caso contrario, guarda en nodeIDorDataOffset el ID del nodo padre y devuele "false".

```
bool _findKey(const char *key_id, FILE *f, int node_id, int *nodeIDOrDataOffset)
```

Función auxiliar de findKey que se encarga de la recursividad de la búsqueda binaria. En este caso key_id identifica el book_id del libro buscado. El resto de variables y retornos son los mismos que la función findKey que la encapsula (a _findKey).

```
bool addIndexEntry(char *book_id, int bookOffset, char const *indexName)
```

Dicha función se encarga de añadir al fichero que contiene los índices un nuevo nodo con el book_id y bookOffset, los cuales recibe como argumentos. Para ello tiene en cuenta diversas cosas:

La primera de todas es la posición que debe de ocupar en la estructura del árbol binario representado por el índice . Para ello se apoya en findKey para obtener el ID del nodo padre, el cual actualiza, y posteriormente itera la lista de nodos borrados hasta encontrar el primer hueco libre en el cual introducirlo (sino hay hueco se introduce al final). Tras la escritura de dicho nodo en el índice procede a actualizar la lista de nodos borrados. Finalmente, cabe destacar que si dicho book_id ya se encuentra en la base de datos, por ser una clave primaria se rechaza y se devuelve "false", sino se devuelve "true".

```
bool addTableEntry(Book *book, const char *tableName, const char *indexName)
```

La última función orientada a la gestión de nuestra base de datos es addTableEntry. Esta función se encarga de añadir al fichero de datos un libro propiamente dicho, y actualizar el fichero de índices. Igual que addIndexTable, la función recorre los nodos borrados hasta encontrar un hueco libre en el que introducir el libro, para ello se sirve de los indicadores de longitud que contienen los registros. Al encontrar un hueco, si sobra suficiente espacio para añadir otro libro, actualiza la lista de registros borrados de manera que se tenga en cuenta el nuevo hueco y se actualiza su longitud; sino solo actualiza la lista de borrados. Igualmente, si es demasiado largo el título del libro, se introduce al final.

Dicha función termina por llamar a addIndexEntry y añadir el libro al índice. Como en addIndexEntry, se devuelve "false" si la clave ya estaba contenida y "true" en caso de éxito.

Comentarios:

También se ha profundizado en el aprendizaje de las funciones de escritura en c:

- *fread*
- *fwrite*
- *fseek*
- *ftell*

Todas estas funciones se utilizan para la escritura y lectura en binario en los ficheros. Aunque para abrir los ficheros nos apoyamos en *fopen* y los correspondientes parámetros:

"rb", "rb+", "wb", "wb+" y "ab", "ab+"

Menú

Para la realización de menú se ha aplicado una metodología similar a la práctica anterior, es decir, se ha intentado modular lo máximo posible para una mayor flexibilidad. Primero se han realizado funciones de apoyo, como MainMenu, que te imprime el menú con el que interacciona el usuario, y get_option, que devuelve la elección del usuario con el respectivo control de input; y finalmente el main con el programa principal.

```
static void MainMenu()
```

```
int get_option(int min, int max)
```

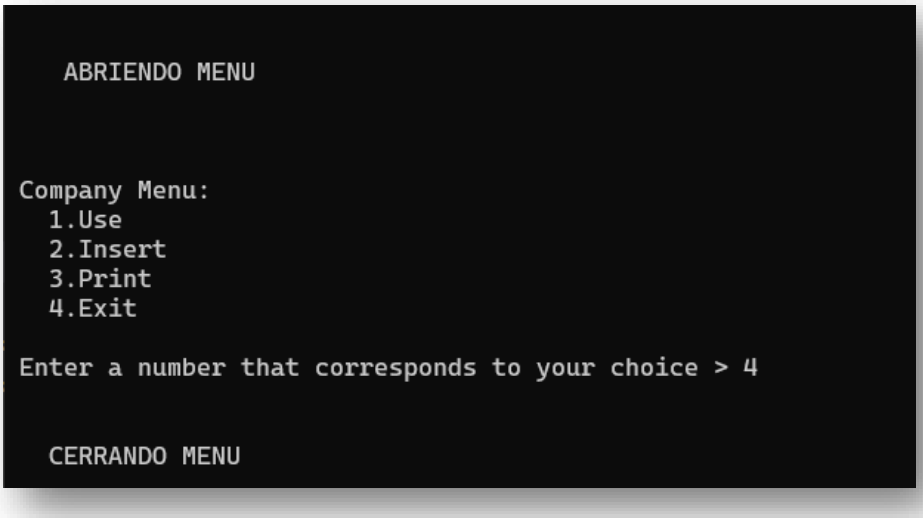
En cuanto al uso del menú, primero es necesario introducir la base de datos que se quiere utilizar (escribiendo exclusivamente el nombre de la DB, y no nombre.dat), pues si no al intentar usar cualquier otra función salta un mensaje de error.

Por otro lado, a la hora de introducir un nuevo libro, la longitud del book_id deber ser exactamente 4, necesario para introducirlo correctamente en la base de datos. En caso de ya estar contenido imprime un mensaje de error, aunque independientemente de que sea exitosa la operación el usuario vuelve al menú.

En cuanto a la impresión del índice de la base de datos, el menú pide especificar el número máximo de niveles a imprimir, empezando por el 0 para el nodo raíz, y con la opción de introducir el -1 para imprimir el árbol en su totalidad. (Si se introduce negativo o una cadena de caracteres que por ente no es un número, por convenio solo imprimirá la raíz).

Cabe destacar que es posible cambiar de base de datos durante la ejecución, usando Use para especificar una DB nueva. Finalmente, para salir del programa basta con introducir 4.

A continuación se aprecia un ejemplo del menú:



Tests

A la hora de comprobar el correcto funcionamiento de las funciones, se ha optado por la metodología TDD, como ya se ha comentado. Para las funciones iniciales se han usado los test proporcionados, los cuales se ejecutan sin ningún problema:

```
* replaceExtensionByIdx: OK
* checkcreateTablecreateIndex: OK
* checkcreateTable: OK
* checkcreateTablecreateIndex: OK
* checkFindKey: OK
```

Como se observa, también se ha usado checkreplaceExtensionByInx y se ha procedido a su correcta comprobación. En cuanto a checkaddTableIndex, aparte de ejecutar el test completo (que se ha terminado entregando tal y como había sido proporcionado), también se modificó para poder observar la evolución de los nodos añadidos.

```
-----Original tree-------
  MAR2 (0): 4
   l MAR0 (2): 37
      l BAR0 (5): 88
         r CAR5 (9): 157
      r MAR1 (6): 105
   r VAR1 (3): 53
      l PAR2 (7): 121
         r RAR2 (10): 174
      r WAR3 (1): 21
-----after adding VAR2-------
  MAR2 (0): 4
   l MAR0 (2): 37
      l BAR0 (5): 88
         r CAR5 (9): 157
      r MAR1 (6): 105
   r VAR1 (3): 53
      l PAR2 (7): 121
         r RAR2 (10): 174
      r WAR3 (1): 21
         l VAR2 (4): 123
```

```
-----after adding VAR3-------
  MAR2 (0): 4
   l MAR0 (2): 37
      l BAR0 (5): 88
         r CAR5 (9): 157
      r MAR1 (6): 105
   r VAR1 (3): 53
      l PAR2 (7): 121
         r RAR2 (10): 174
      r WAR3 (1): 21
         l VAR2 (4): 123
            r VAR3 (8): 321
-----after adding VAR4-------
  MAR2 (0): 4
   l MAR0 (2): 37
      l BAR0 (5): 88
         r CAR5 (9): 157
      r MAR1 (6): 105
   r VAR1 (3): 53
      l PAR2 (7): 121
         r RAR2 (10): 174
      r WAR3 (1): 21
         l VAR2 (4): 123
            r VAR3 (8): 321
               r VAR4 (11): 678
--------------------------------
*checking AddIndexEntry: OK
```

Finalmente, el ultimo test implementado ha sido addTableEntry, desarrollado por nosotros con la finalidad de observar y comprobar que se introducían correctamente los nuevos libros a la base de datos. Para ello, se ha modificado tester.c para poder añadirle un argumento adicional que indique cuantos libros se le quieren añadir, con un mínimo de 1 y un máximo de 4, y así poder estudiar la evolución de la DB.

```
./tester 2
```

```
-------------Checking AddTableEntry----------------

Error: Book with id=MAR2 already in DBEXPECTED ERROR STATUS: OK
-----Original tree-------
  MAR2 (0): 4
   l MAR0 (2): 37
      l BAR0 (5): 88
         r CAR5 (9): 157
      r MAR1 (6): 105
   r VAR1 (3): 53
      l PAR2 (7): 121
         r RAR2 (10): 174
      r WAR3 (1): 21
-----after adding VAR2-------
  MAR2 (0): 4
   l MAR0 (2): 37
      l BAR0 (5): 88
         r CAR5 (9): 157
      r MAR1 (6): 105
   r VAR1 (3): 53
      l PAR2 (7): 121
         r RAR2 (10): 174
      r WAR3 (1): 21
         l VAR2 (4): 71
-----after adding SPAC-------
  MAR2 (0): 4
   l MAR0 (2): 37
      l BAR0 (5): 88
         r CAR5 (9): 157
      r MAR1 (6): 105
   r VAR1 (3): 53
      l PAR2 (7): 121
         r RAR2 (10): 174
            r SPAC (8): 139
      r WAR3 (1): 21
         l VAR2 (4): 71
-----------------------------------------------
*checking AddTableEntry: OK
```

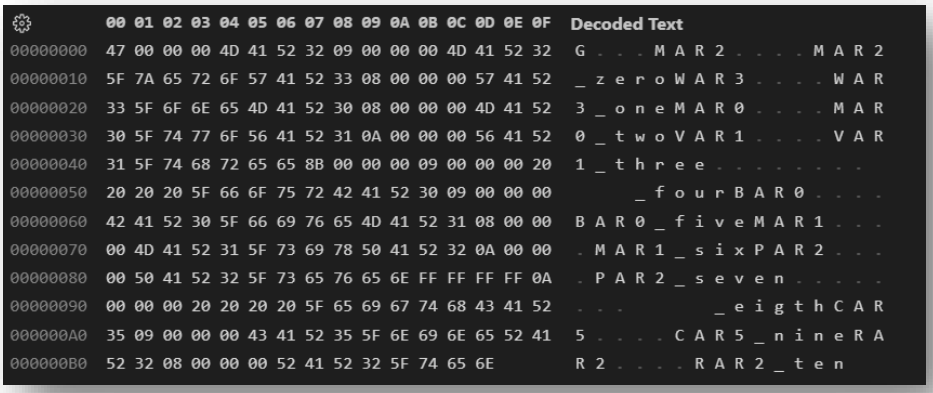
Como se aprecia en la anterior imagen, comprueba que al intentar añadir un libro que ya existe te devuelve un mensaje de error. Luego imprime el árbol original y más adelante el árbol con los nuevos libros. Aunque no se muestra explícitamente en la salida del código, el test si que comprueba que los offset y los registros borrados se actualizan debidamente.

A continuación tenemos el código de testeo para SPAC:

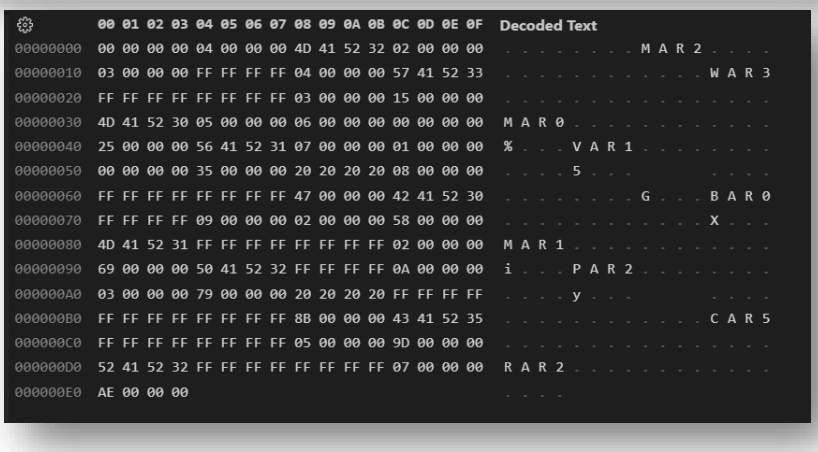
```
/* add another new book using an existing deleted book */
    memcpy(newbook.book_id, "SPAC", 4);
    newbook.title="E";
    newbook.title_len=1;
    offset = 139; /*8b en HEX*/
    deleted = 148; /*94 en HEX*/
    result = addTableEntry(&newbook, tableName, indexName);
    findKey(newbook.book_id, indexName, &nodeIDOrDataOffset);
    printf("-----after adding SPAC-------\n");
    printTree(10, indexName);
    if (nodeIDOrDataOffset!=offset){
        fprintf(stderr,
                "Error in addTableEntry, inserted offset should be %d"
                " but I get %d", offset, nodeIDOrDataOffset);
        exit(EXIT_FAILURE);
    }
    /* get deleted 148 (first deleted) */
    fp = fopen(tableName, "r");
    fread(&tmp, sizeof(int), 1, fp);
    if (tmp!=deleted){
        fprintf(stderr,
                "Error in addTableEntry, first int in file should be %d, that
                 is %X in hex, but is %d", deleted, deleted, tmp);
        exit(EXIT_FAILURE);
    }
    /*get deleted -1 (second deleted) */
    fseek(fp,deleted,SEEK_SET);
    fread(&tmp, sizeof(int), 1, fp);
    if (tmp!=-1){
        fprintf(stderr,
                "Error in addTableEntry, pointer to second deleted space in
                file should be %d" " but is %d", -1, tmp);
        exit(EXIT_FAILURE);
    }

    fclose(fp);
```

Mediante la introducción de otros 2 libros (4 en total), se comprueba que los registros borrados se actualizan correctamente y que se introducen los nuevos libros en aquellos con suficiente espacio. Al poder especificar el numero de libros a insertar, y apoyándonos en el lector hexadecimal, se puede visualizar la evolución del algoritmo:

1)

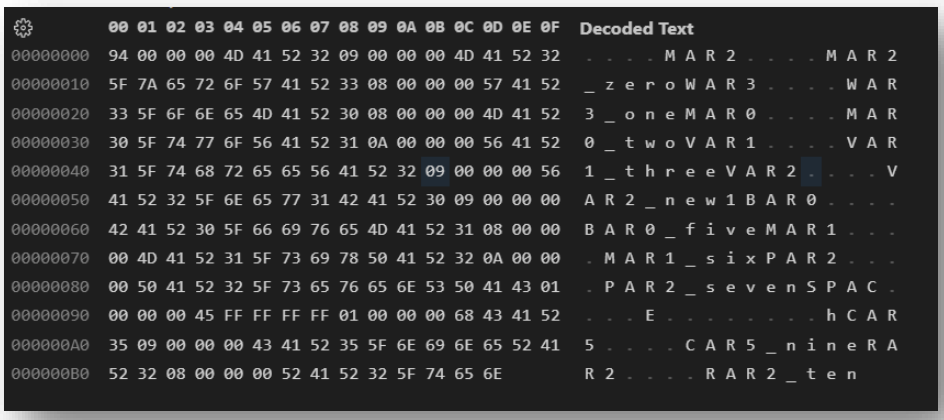


	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded Text
00000000	47 00 00 00 4D 41 52 32 09 00 00 00 4D 41 52 32	G . . . M A R 2 . . . . M A R 2
00000010	5F 7A 65 72 6F 57 41 52 33 08 00 00 00 57 41 52	_ z e r o W A R 3 . . . . W A R
00000020	33 5F 6F 6E 65 4D 41 52 30 08 00 00 00 4D 41 52	3 _ o n e M A R 0 . . . . M A R
00000030	30 5F 74 77 6F 56 41 52 31 0A 00 00 00 56 41 52	0 _ t w o V A R 1 . . . . V A R
00000040	31 5F 74 68 72 65 65 8B 00 00 00 09 00 00 00 20	1 _ t h r e e . . . . .
00000050	20 20 20 5F 66 6F 75 72 42 41 52 30 09 00 00 00	_ f o u r B A R 0 . . . .
00000060	42 41 52 30 5F 66 69 76 65 4D 41 52 31 08 00 00	B A R 0 _ f i v e M A R 1 . . .
00000070	00 4D 41 52 31 5F 73 69 78 50 41 52 32 0A 00 00	. M A R 1 _ s i x P A R 2 . . .
00000080	00 50 41 52 32 5F 73 65 76 65 6E FF FF FF FF 0A	. P A R 2 _ s e v e n . . . . .
00000090	00 00 00 20 20 20 20 5F 65 69 67 74 68 43 41 52	. . . _ e i g t h C A R
000000A0	35 09 00 00 00 43 41 52 35 5F 6E 69 6E 65 52 41	5 . . . . C A R 5 _ n i n e R A
000000B0	52 32 08 00 00 00 52 41 52 32 5F 74 65 6E	R 2 . . . . R A R 2 _ t e n

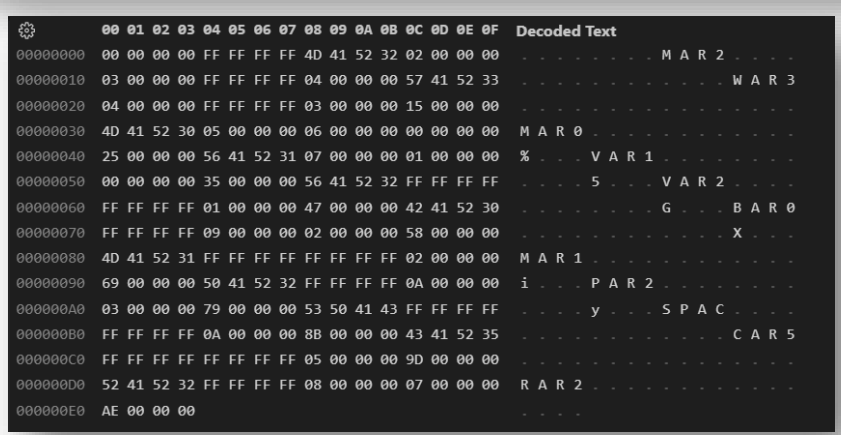


	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded Text
00000000	00 00 00 00 04 00 00 00 4D 41 52 32 02 00 00 00	. . . . . M A R 2 . . . .
00000010	03 00 00 00 FF FF FF FF 04 00 00 00 57 41 52 33	. . . . . W A R 3
00000020	FF FF FF FF FF FF FF FF 03 00 00 00 15 00 00 00	. . . . .
00000030	4D 41 52 30 05 00 00 00 06 00 00 00 00 00 00 00	M A R 0 . . . . .
00000040	25 00 00 00 56 41 52 31 07 00 00 00 01 00 00 00	% . . . V A R 1 . . . . .
00000050	00 00 00 00 35 00 00 00 20 20 20 20 08 00 00 00	. . . . 5 . . . . .
00000060	FF FF FF FF FF FF FF FF 47 00 00 00 42 41 52 30	. . . . . G . . . B A R 0
00000070	FF FF FF FF 09 00 00 00 02 00 00 00 58 00 00 00	. . . . . X . . .
00000080	4D 41 52 31 FF FF FF FF FF FF FF FF 02 00 00 00	M A R 1 . . . . .
00000090	69 00 00 00 50 41 52 32 FF FF FF FF 0A 00 00 00	i . . . P A R 2 . . . . .
000000A0	03 00 00 00 79 00 00 00 20 20 20 20 FF FF FF FF	. . . . y . . . . .
000000B0	FF FF FF FF FF FF FF FF 8B 00 00 00 43 41 52 35	. . . . . C A R 5
000000C0	FF FF FF FF FF FF FF FF 05 00 00 00 9D 00 00 00	. . . . .
000000D0	52 41 52 32 FF FF FF FF FF FF FF FF 07 00 00 00	R A R 2 . . . . .
000000E0	AE 00 00 00	. . . .

2)

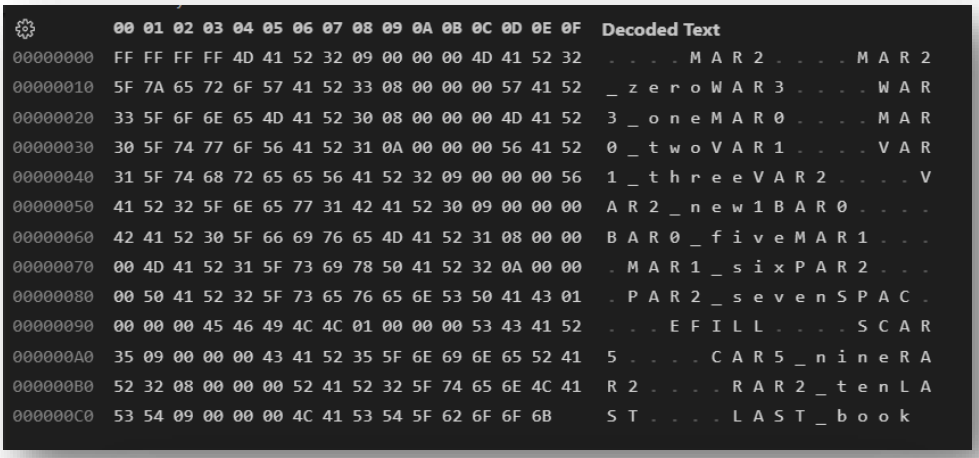


	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded Text
00000000	94 00 00 00 4D 41 52 32 09 00 00 00 4D 41 52 32	. . . . M A R 2 . . . . M A R 2
00000010	5F 7A 65 72 6F 57 41 52 33 08 00 00 00 57 41 52	_ z e r o W A R 3 . . . . W A R
00000020	33 5F 6F 6E 65 4D 41 52 30 08 00 00 00 4D 41 52	3 _ o n e M A R 0 . . . . M A R
00000030	30 5F 74 77 6F 56 41 52 31 0A 00 00 00 56 41 52	0 _ t w o V A R 1 . . . . V A R
00000040	31 5F 74 68 72 65 65 56 41 52 32 09 00 00 00 56	1 _ t h r e e V A R 2 V
00000050	41 52 32 5F 6E 65 77 31 42 41 52 30 09 00 00 00	A R 2 _ n e w 1 B A R 0 . . . .
00000060	42 41 52 30 5F 66 69 76 65 4D 41 52 31 08 00 00	B A R 0 _ f i v e M A R 1 . . .
00000070	00 4D 41 52 31 5F 73 69 78 50 41 52 32 0A 00 00	. M A R 1 _ s i x P A R 2 . . .
00000080	00 50 41 52 32 5F 73 65 76 65 6E 53 50 41 43 01	. P A R 2 _ s e v e n S P A C .
00000090	00 00 00 45 FF FF FF FF 01 00 00 00 68 43 41 52	. . . E . . . . . h C A R
000000A0	35 09 00 00 00 43 41 52 35 5F 6E 69 6E 65 52 41	5 . . . . C A R 5 _ n i n e R A
000000B0	52 32 08 00 00 00 52 41 52 32 5F 74 65 6E	R 2 . . . . R A R 2 _ t e n

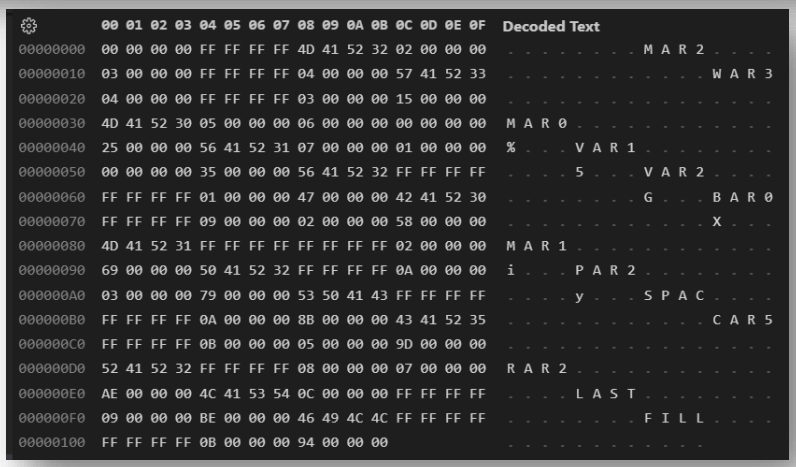


	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded Text
00000000	00 00 00 00 FF FF FF FF 4D 41 52 32 02 00 00 00	. . . . . M A R 2 . . . .
00000010	03 00 00 00 FF FF FF FF 04 00 00 00 57 41 52 33	. . . . . W A R 3
00000020	04 00 00 00 FF FF FF FF 03 00 00 00 15 00 00 00	. . . . .
00000030	4D 41 52 30 05 00 00 00 06 00 00 00 00 00 00 00	M A R 0 . . . . .
00000040	25 00 00 00 56 41 52 31 07 00 00 00 01 00 00 00	% . . . V A R 1 . . . . .
00000050	00 00 00 00 35 00 00 00 56 41 52 32 FF FF FF FF	. . . . 5 . . . V A R 2 . . . .
00000060	FF FF FF FF 01 00 00 00 47 00 00 00 42 41 52 30	. . . . . G . . . B A R 0
00000070	FF FF FF FF 09 00 00 00 02 00 00 00 58 00 00 00	. . . . . X . . .
00000080	4D 41 52 31 FF FF FF FF FF FF FF FF 02 00 00 00	M A R 1 . . . . .
00000090	69 00 00 00 50 41 52 32 FF FF FF FF 0A 00 00 00	i . . . P A R 2 . . . . .
000000A0	03 00 00 00 79 00 00 00 53 50 41 43 FF FF FF FF	. . . . y . . . S P A C . . . .
000000B0	FF FF FF FF 0A 00 00 00 8B 00 00 00 43 41 52 35	. . . . . C A R 5
000000C0	FF FF FF FF FF FF FF FF 05 00 00 00 9D 00 00 00	. . . . .
000000D0	52 41 52 32 FF FF FF FF 08 00 00 00 07 00 00 00	R A R 2 . . . . .
000000E0	AE 00 00 00	. . . .

3)



	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded Text
00000000	FF FF FF FF 4D 41 52 32 09 00 00 00 4D 41 52 32	....MAR2....MAR2
00000010	5F 7A 65 72 6F 57 41 52 33 08 00 00 00 57 41 52	_zeroWAR3....WAR
00000020	33 5F 6F 6E 65 4D 41 52 30 08 00 00 00 4D 41 52	3_oneMAR0....MAR
00000030	30 5F 74 77 6F 56 41 52 31 0A 00 00 00 56 41 52	0_twoVAR1....VAR
00000040	31 5F 74 68 72 65 65 56 41 52 32 09 00 00 00 56	1_threeVAR2....V
00000050	41 52 32 5F 6E 65 77 31 42 41 52 30 09 00 00 00	AR2_new1BAR0....
00000060	42 41 52 30 5F 66 69 76 65 4D 41 52 31 08 00 00	BAR0_fiveMAR1...
00000070	00 4D 41 52 31 5F 73 69 78 50 41 52 32 0A 00 00	.MAR1_sixPAR2...
00000080	00 50 41 52 32 5F 73 65 76 65 6E 53 50 41 43 01	.PAR2_sevenSPAC.
00000090	00 00 00 45 46 49 4C 4C 01 00 00 00 53 43 41 52	...EFILL....SCAR
000000A0	35 09 00 00 00 43 41 52 35 5F 6E 69 6E 65 52 41	5....CAR5_nineRA
000000B0	52 32 08 00 00 00 52 41 52 32 5F 74 65 6E 4C 41	R2....RAR2_tenLA
000000C0	53 54 09 00 00 00 4C 41 53 54 5F 62 6F 6F 6B	ST....LAST_book



	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded Text
00000000	00 00 00 00 FF FF FF FF 4D 41 52 32 02 00 00 00	........MAR2....
00000010	03 00 00 00 FF FF FF FF 04 00 00 00 57 41 52 33	............WAR3
00000020	04 00 00 00 FF FF FF FF 03 00 00 00 15 00 00 00	....
00000030	4D 41 52 30 05 00 00 00 06 00 00 00 00 00 00 00	MAR0............
00000040	25 00 00 00 56 41 52 31 07 00 00 00 01 00 00 00	%...VAR1........
00000050	00 00 00 00 35 00 00 00 56 41 52 32 FF FF FF FF	....5...VAR2....
00000060	FF FF FF FF 01 00 00 00 47 00 00 00 42 41 52 30	........G...BAR0
00000070	FF FF FF FF 09 00 00 00 02 00 00 00 58 00 00 00	............X...
00000080	4D 41 52 31 FF FF FF FF FF FF FF FF 02 00 00 00	MAR1............
00000090	69 00 00 00 50 41 52 32 FF FF FF FF 0A 00 00 00	i...PAR2........
000000A0	03 00 00 00 79 00 00 00 53 50 41 43 FF FF FF FF	....y...SPAC....
000000B0	FF FF FF FF 0A 00 00 00 8B 00 00 00 43 41 52 35	............CAR5
000000C0	FF FF FF FF 0B 00 00 00 05 00 00 00 9D 00 00 00	....
000000D0	52 41 52 32 FF FF FF FF 08 00 00 00 07 00 00 00	RAR2............
000000E0	AE 00 00 00 4C 41 53 54 0C 00 00 00 FF FF FF FF	....LAST........
000000F0	09 00 00 00 BE 00 00 00 46 49 4C 4C FF FF FF FF	........FILL....
00000100	FF FF FF FF 0B 00 00 00 94 00 00 00	....

Dichas imágenes muestran el estado inicial, intermedio y fianal tanto del fichero de datos como del índice. Lo que corresponde al test completo:

```
-----Original tree-------
  MAR2 (0): 4
   l MAR0 (2): 37
      l BAR0 (5): 88
         r CAR5 (9): 157
      r MAR1 (6): 105
   r VAR1 (3): 53
      l PAR2 (7): 121
         r RAR2 (10): 174
      r WAR3 (1): 21
-----after adding VAR2-------
  MAR2 (0): 4
   l MAR0 (2): 37
      l BAR0 (5): 88
         r CAR5 (9): 157
      r MAR1 (6): 105
   r VAR1 (3): 53
      l PAR2 (7): 121
         r RAR2 (10): 174
      r WAR3 (1): 21
         l VAR2 (4): 71
```

```
-----after adding SPAC-------
  MAR2 (0): 4
   l MAR0 (2): 37
      l BAR0 (5): 88
         r CAR5 (9): 157
      r MAR1 (6): 105
   r VAR1 (3): 53
      l PAR2 (7): 121
         r RAR2 (10): 174
            r SPAC (8): 139
      r WAR3 (1): 21
         l VAR2 (4): 71
-----after adding LAST-------
  MAR2 (0): 4
   l MAR0 (2): 37
      l BAR0 (5): 88
         r CAR5 (9): 157
            r LAST (11): 190
      r MAR1 (6): 105
   r VAR1 (3): 53
      l PAR2 (7): 121
         r RAR2 (10): 174
            r SPAC (8): 139
      r WAR3 (1): 21
         l VAR2 (4): 71
```

```
-----after adding FILL-------
  MAR2 (0): 4
   l MAR0 (2): 37
      l BAR0 (5): 88
         r CAR5 (9): 157
            r LAST (11): 190
               l FILL (12): 148
      r MAR1 (6): 105
   r VAR1 (3): 53
      l PAR2 (7): 121
         r RAR2 (10): 174
            r SPAC (8): 139
      r WAR3 (1): 21
         l VAR2 (4): 71
--------------------------------
*checking AddTableEntry: OK
```

Makefile

En cuanto a la compilación de los ejecutables, el makefile se encarga de compilar todos los archivos y tests necesarios para el tester mediante el comando "make all". Por otro lado, también se ha añadido el comando "make run" para compilar y ejecutar directamente el menú.

Para comprobar que no hay perdidas de memoria se ha ejecutado tanto el tester como el menú con valgrind, para los cuales también se han creado los shortcuts, "test_val" y "menu_val" resprectivamente.

```
run: $(MENU)
    @./menu

test_val:
    @valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes
./tester

menu_val:
    @valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes
./menu

clean :
    rm -f *.o core $(EXE)

clear:
    rm -f -v $(MENU) $(EXE)
```

Por último, hemos modificado algunos de los test proporcionados para eliminar las fugas de memoria que tenían. De esta manera tanto el tester como el menú se ejecutan sin perdidas:

```
*checking AddTableEntry: OK
==151==
==151== HEAP SUMMARY:
==151==     in use at exit: 0 bytes in 0 blocks
==151==   total heap usage: 253 allocs, 253 frees, 341,384 bytes allocated
==151==
==151== All heap blocks were freed -- no leaks are possible
==151==
==151== For lists of detected and suppressed errors, rerun with: -s
==151== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

```
  CERRANDO MENU

==153==
==153== HEAP SUMMARY:
==153==     in use at exit: 0 bytes in 0 blocks
==153==   total heap usage: 19 allocs, 19 frees, 34,535 bytes allocated
==153==
==153== All heap blocks were freed -- no leaks are possible
==153==
==153== For lists of detected and suppressed errors, rerun with: -s
==153== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Conclusión

En conclusión, el desarrollo de la practica nos ha instruido tanto en la implementación de bases de datos como en el desarrollo de proyectos informáticos, además de enseñarnos nuevas metodologías y ampliar nuestro conocimiento respecto a la lectura y escritura de ficheros.

Participantes:

David Brenchley Uriol

Javier San Andrés de Pedro