UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

FACULTAD DE INGENIERIA

AVE PARA WINDOWS
MANEJADOR DE ARCHIVOS.

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A N:

VICTOR MANUEL CAMACHO VELAZQUEZ
EDUARDO CANSECO OSORIO II
ARTURO PARDO BELLO

DIRECTOR DE TESIS:
Raymundo Hugo Rangel Gutiérrez

MEXICO, D.F. JUNIO DE 1992

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE GENERAL

INTRODUCCION

La risa de los dioses hace naufragar a quien intente proclamarse juez en el campo de la verdad y del conocimiento.

INTRODUCCION

Actualmente en México ya existen aplicaciones que de alguna manera motivan el desarrollo de software nacional, sin embargo, lo anterior se satisface sólo de manera parcial, ya que es común que se elaboren programas para cubrir reducidas necesidades pero, nunca se realizan con una visión general para el mercado de software.

Algunas veces nos hemos preguntado acerca de el por qué en nuestro país no se desarrolla una industria similar a la de otros, capaz de lanzar productos de software que fueran competitivos a nivel comercial. La respuesta, tal vez, se deba a la inercia que existe para realizar el esfuerzo y gasto económico que esa actividad conlleva pues con frecuencia se escucha el comentario de que si un producto ya existe, ¿Porqué tomarse la molestia en desarrollar uno similar?.

Tal vez lo antes expresado tenga algo de verdad, sin embargo, creemos que siendo el diseño la esencia de la ingeniería estamos obligados y ser capaces de elaborar sistemas de software tan buenos como los extranjeros, aún más : planearlos de acuerdo a los requerimientos futuros que las necesidades demandarán y no quedarnos como simples usuarios o espectadores maravillados ante la "magia" desplegada que algunas veces nos entretiene también.

Entonces, para nosotros, la pregunta ¿Cómo hacerlo? fue sin lugar a duda el acicate, siempre presente, en el desarrollo del sistema ha presentar; si bien modesto; tiene la esencia de haber sido construido *"desde abajo"*, con esto queremos decir que no es producto o una aplicación de un "paquete" preexistente.

Concretando, los motivos que nos indujeron a elaborar este trabajo de tesis fueron por un lado, hacer a un lado ese velo, aunque sea un poco, para ver lo que hay detrás de un paquete de software; por el otro, al involucrar una utilería reciente como lo es WINDOWS, ejemplificar alguna de las variadas y posibles tendencias en los ambientes de PC's; finalmente lo esperado: terminar lo que comenzamos.

PRESENTACION

La experiencia mas hermosa que tenemos a nuestro alcance es el misterio.

PRESENTACION

Antes de dar paso a la pormenorización del sistema AVE, es conveniente relatar a grandes rasgos la evolución que tuvo la concepción del sistema, así como de nuestras vicisitudes con los aperos de software de que nos valimos.

Siempre que se tiene la intención de crear algo, no necesariamente novedoso, sin importar el campo del quehacer humano, se hace evidente al cabo de pocas vueltas que la gama de aplicaciones potenciales es amplia y que sobrepasa en ocasiones nuestra experiencia cotidiana, además, la variedad de aplicaciones en un sector específico nos resulta desconocida; el área de aplicaciones con computadora no es la excepción, pues, por un lado, la finalidad de la carrera de computación en la Facultad de Ingeniería no pretende poner al tanto sobre tal o cual paquete de software; sus objetivos son distintos; por otro lado, no nos desenvolvemos en un ambiente de aplicaciones que nos permita el acceso al manejo de programas de aplicación específica; en el mejor de los casos, nuestro entender sobre tal o cual software tiene su origen en el conocimiento de algunas facilidades de dominio público que en ciertos momentos nos han auxiliado en tareas escolares; no obstante, nuestra comprensión, diríamos no cabal, se circunscribe a lo más a dos o tres utilerías afines. Mucho de ese conocimiento, que en realidad no lo es, proviene de revistas de computación donde se hace propaganda a las bondades cada vez mas elocuentes de los paquetes de software; en otras palabras, dicha propaganda no es más que una exaltada referencia que definitivamente no conduce a una apreciación funcional del software, ni mucho menos visible es su estructura algorítmica y organizacional a nivel modular, como a veces se sospecha. En algunos casos, los menos, se

vislumbra apenas un rasgo que conduce, a modo especulativo, al cómo funciona el producto y cuáles pudieran ser las estructuras de datos básicas del mismo.

Con la perspectiva anterior, resulta difícil para los novatos diseñadores de software pretender en una primera experiencia equiparar y ya no digamos superar alguna utilería de vanguardia. Lo antes expresado se fundamenta en cierto grado por la carencia de conocimientos específicos necesarios para poder encarar un proyecto de tesis, mas no por incapacidad; otro factor plausible es el tiempo estimado para terminar un producto, pues asentimos que debe haber un intervalo razonable a fin de lograr un prototipo acorde a la espiral evolutiva del software, es decir, no permitir un tiempo de desarrollo exagerado que arroje un producto caduco en términos conceptuales; pues no resulta difícil el darse cuenta que en el mundo del software lo que hoy es la octava maravilla, en poco tiempo deja de serlo; con la consecuente sensación de haber elaborado un producto a destiempo.

Conscientes de lo anterior, pensamos que un administrador de archivos podría darnos la oportunidad de proyectar nuestras inquietudes, que con la guía adecuada se enfocaron y cristalizaron en este trabajo.

El objetivo de fondo fue siempre darle al producto la posibilidad de ser utilizado por usuarios no especializados en computación, es decir, un paquete sencillo donde su manejo fuera casi obvio (tal vez no así su aplicación) de manera que, la idea de partida fue tomar como referencia algún manejador de archivos que fuera popular y de fácil acceso, nuestra búsqueda recayó sobre el DBase III de entonces. Acto seguido se comenzó a conocer su manejo a modo de familiarizarse y dar inicio a reflexiones tendientes a cómo poder crear un software con características parecidas, inclusive la presentación con ventanas la mantuvimos vigente.

Antes que detalláramos las estructuras de datos necesarias y las habilidades a dotar al sistema, se tuvo que pensar en un lenguaje de desarrollo lo suficiente poderoso y apto para nuestra aplicación en perspectiva. Tomamos en consideración dos lenguajes: "Pascal" por su amplia divulgación y aceptable conocimiento por parte nuestra y el "C" pues, reúne cualidades ideales tales como la de generar un código reducido y la velocidad de ejecución altamente satisfactoria, ad hoc a nuestros requerimientos, además, en ese entonces comenzaba a ganar adeptos. Nos decidimos por el "C", no sólo por las razones antes esgrimidas sino también porque nos enteramos que mucho del software de nuevo desarrollo se estaba implementando en "C" al igual que algunos sistemas operativos de amplia divulgación. El caso es que no sabíamos ni pizca de "C", de modo que al tiempo en que se comenzaron a planear las estructuras de datos del sistema, estudiamos tal novedoso y misterioso lenguaje.

Comenzamos a desarrollar los programas primordiales del sistema, entre ellos los relativos a las ventanas de presentación, observamos los primeros logros y no quedamos muy convencidos pues, su despliegue era muy lento, así que para mejorar tal detrimento, se recurrió a ciertas técnicas de programación en lenguaje ensamblador 8086; por supuesto, esa fue la solución: los desplegados iban y venían en un abrir y cerrar de ojos. Por otro lado, los progresos en otros rubros comenzaron a menguar pues, muchas de las estructuras de datos de regular complejidad, si bien, eran aceptadas por el compilador, no funcionaban, se pensó que tales anomalías podrían tener su origen, en parte a que no conocíamos de manera cabal al lenguaje y sus "trucos" con razonable dominio, sin embargo, el compilador pronto comenzó a dar muestra de sus limitaciones, algunas de ellas poco creíbles y que consumieron demasiado tiempo en ser detectadas, mientras tanto el tiempo transcurría en pruebas por demás decepcionantes y nada fructíferas que a largo plazo se reflejaron en la moda de visualizar la información a través de ventanas amigables; esa se había

vuelto obsoleta. Ante tal deterioro hubo que revalorar lo hecho y lo usado, es decir lo alcanzado y la herramienta que nos había permitido tal logro; decidimos entonces, en primer lugar, darle un giro al aspecto visual del sistema, sustituyendo la presentación de ventanas por una más acorde a las tendencias de hoy día: el manejo de aplicaciones bajo la administración de Windows. Este enfoque no únicamente mejora el atractivo visual que se le pueda adherir a un paquete de software, sino también lo vuelve amistoso. Por supuesto, tales mejoras implicaron una revisión concienzuda de todo el código fuente, pues Windows requiere la generación de código derivado de alguno de los "C Microsoft Compilers", versiones de la cinco en adelante, en este caso se empleó la versión seis.

Finalmente, queremos expresar que este trabajo lo percibimos como un experimento, por demás largo, y, en ciertos instantes desesperante, donde se puso a prueba tanto organización de equipo, conocimientos y demasiada paciencia; no pretendiendo en ningún momento originar un producto de amplios cauces mercantiles pues, las circunstancias para satisfacer tal objetivo, no creemos sean las idóneas ni los fines de una tesis; simplemente como una actividad generadora de experiencia preparatoria para el ineludible devenir de actividades económico-sociales; simplemente algo ejemplificador de lo nuevo y promisorio, y claro está perfectible, sin lugar a dudas, respecto a sus alcances, con la esperanza, si acaso, de poder servir como base de ulteriores desarrollos.

CAPITULO I

LA ESTRUCTURA DE DATOS FUNDAMENTAL

La alegría de mirar y comprender ...
... es el don mas hermoso de la naturaleza.

I. LA ESTRUCTURA DE DATOS FUNDAMENTAL

Con el objeto de llevar a cabo un seguimiento de los archivos de datos que están relacionados con el sistema, es necesario contar con una estructura de datos adecuada para tal finalidad.

Así, se considera que un archivo sería la estructura fundamental cuya función primordial será la de mantener las características propias de cada uno de los archivos de datos, esto es, contener la estructura que fija el tipo de la información que contendrá cada uno de los archivos de datos.

Además, la estructura fundamental contiene información acerca del total de archivos de datos dados de alta o baja.

El archivo al cual se hace referencia contiene la estructura fundamental y es básicamente un *Archivo Descriptor de Archivos* al que llamaremos por abreviar *ADA*.

Precisando acerca de la organización que presenta ADA se tiene lo siguiente: es un archivo secuencial constituído por registros de tamaño fijo de 1024 bytes.

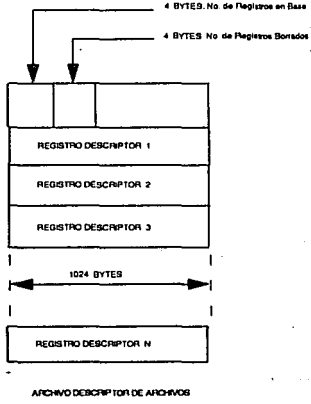
El ADA está dividido en dos partes : La primera que corresponde al registro inicial, la hemos designado cabeza de lista, la cual está presente aún cuando no haya archivos de datos dados de alta. De este registro sólo son utilizados ocho bytes, los primeros cuatro de ellos mantiene el número total de estructuras de archivos de datos que han sido dadas de alta y que pueden ser reconocidas por el sistema. El segundo

cuarteto de bytes contiene otro número que identifica la posición, dentro de ADA del último descriptor de un archivo de datos que ha sido dado de baja.

Los ocho bytes antes comentados son alterados conforme se crean y se eliminan descriptores que definen archivos, pero aún cuando no haya descriptores, inicialmente mantienen cada uno un valor de cero.

La segunda porción del ADA que principia a partir del byte número nueve, alberga información asociada exclusivamente con las características de los datos de los archivos.

La siguiente figura nos ilustra de manera general la estructura del Archivo Descriptor de Archivos(ADA).



ARCHIVO DESCRIPTOR DE ARCHIVOS

Ahora bien, lo distintivo del ADA es la manera en que son almacenadas y manejadas las características descriptivas de los archivos de datos y, es que las manipulaciones son llevadas a cabo a nivel de byte lo cual presenta las siguientes ventajas:

a) El espacio asignado para la descripción de cada archivo se optimiza.

b) Se conoce la posición exacta de almacenamiento de cada componente.

A continuación se describen los diferentes elementos constitutivos que podrán aparecer en la descripción de un archivo:

1.- *Trayectoria:*
Indica la unidad de disco y posibles subdirectorios donde se localiza el archivo de datos.

2.- *Nombre del archivo:*
Es el nombre del archivo incluyendo posible extensión.

3.- *Tipo de archivo:*
El tipo de un archivo puede ser índice o de datos. Tiene una longitud de un byte (No usado en este caso).

4.- *Número de bytes utilizados:*
A cada campo se le asocia una longitud del número máximo de bytes o caracteres que puede almacenar; esta especificación es la suma de las longitudes de todos los campos incluídos en la descripción de un archivo. Tiene cuatro bytes asociados.

5.- *Total de campos definidos en el archivo.*
El número total de campos definidos en el descriptor. Tamaño de un byte.

6.- *Nombre del campo:*

El nombre de cada uno de los campos especificados en el descriptor. Cinco bytes asignados.

7.- *Tipo de campo:*

El tipo de campo puede ser:

-alfanumérico.
-decimal.
-lógico.
-fecha.

Cada tipo de campo tiene asociado un byte.

8.- *Número de tabla:*

La información de un campo puede estar localizada en una tabla de datos. Se le asigna un byte de longitud para identificar la tabla (no es usado en este caso).

9.- *Longitud del campo:*

Cantidad entera que especifica el número de bytes asociados con un campo. Se le asigna un byte.

10.- *Número de decimales:*

En un campo de tipo numérico puede especificar cantidades fraccionarias. Se le asigna un byte.

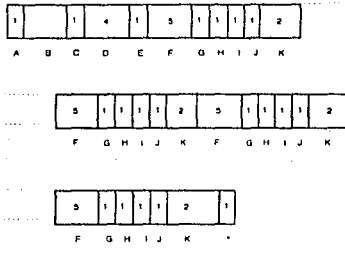
11.- *Inicio de campo:*

Un número que indica el byte donde comienza la información ligada a un campo. Se le asigna dos bytes de longitud.

12.- *Campo liga:*

Este byte contiene un número que hace referencia al siguiente archivo dado de baja. Un byte de longitud.

La figura siguiente muestra gráficamente la estructura del Archivo Descriptor de Archivos (ADA).



CAMPO	DESCRIPCIÓN
A	CONTIENE LA LONGITUD DE LA TRAYECTORIA DEL ARCHIVO MAS LA LONGITUD DEL NOMBRE DEL ARCHIVO MAS 1
B	MANTIENE LA TRAYECTORIA DEL ARCHIVO Y EL NOMBRE DEL ARCHIVO
C	EL TIPO DEL ARCHIVO
D	EL TOTAL DE BYTES DEL REGISTRO
E	TOTAL DE CAMPOS CONTENIDOS EN UN REGISTRO
F	NOMBRE DE UN CAMPO
G	INDICA EL TIPO DE CAMPO
H	ES EL NUMERO DE TABLA A USAR
I	LONGITUD DEL CAMPO
J	CANTIDAD DE DECIMALES A CONSIDERAR EN UN CAMPO NUMERICO
K	INICIO DEL CAMPO
*	CAMPO DE LIGA AL SIGUIENTE REGISTRO VACIO.

UN REGISTRO DEL ADA

Referente a la manera en que el sistema interactua con ADA, brevemente se aclara que cada vez que se ejecuta el sistema se verifica la existencia del archivo ADA.HDR; si no existe se crea y se inicializa el registro de encabezado, en caso contrario, se deja intacto.

Cabe mencionar a manera de aclaración que se cuenta con rutinas que accesan el archivo ADA.HDR. Algunas de ellas sólo recuperan información del ADA, otras lo modifican; tales rutinas son tratadas en los siguientes capítulos.

CAPITULO II

OPERACIONES SOBRE EL ARCHIVO DESCRIPTOR DE ARCHIVOS

No se ganan los hombres con favores, sino con obras.

II. OPERACIONES SOBRE EL ARCHIVO DESCRIPTOR DE ARCHIVOS (ADA)

El archivo ADA.HDR, elemento esencial para el sistema, puede ser modificado si se activa alguna de las operaciones definidas en el menú de "Archivos". Es menester de esta sección describir el funcionamiento de las opciones comprendidas en el menú mencionado.

Cada una de las funciones es independiente y tienen como objetivo interactuar con las estructuras que determinan los tipos de los datos de los archivos que están relacionados con el sistema, mas no con los datos de los archivos.

Ahora bien, el procedimiento para activar cualquier operación ó función es muy simple y basta con posicionar la flecha-cursor sobre la opción deseada y presionar una vez el botón izquierdo del "ratón". Como resultado se tendrá el desplegado de los mensajes y preguntas pertinentes relacionados con la opción elegida.

En lo que resta del capítulo se presentan los objetivos principales y diagramas de flujo de las opciones.

II.1. FUNCION CREAR

Sus objetivos primarios son:

1.- Reservar espacio en el archivo ADA.HDR a fin de poder almacenar un nuevo descriptor de archivo.

2.- Solicitar al usuario los atributos que describirán a un archivo de datos.

3.- Verificar que no haya campos con el mismo nombre.

4.- Checar que cada archivo de datos solo tenga un descriptor asociado.

5.- No permitir la presencia de campos nulos en la definición de un descriptor de archivo.

6.- Actualizar al archivo ADA.HDR cada vez que se crea o elimina un descriptor de archivo.

7.- Salvar un nuevo descriptor.

8.- Regresar el control al sistema.

La figura II.1. muestra el diagrama de flujo de la función crear.

II.2. FUNCION CONSULTAR

Sus fines principales son:

1.- Pedir el nombre del archivo cuya estructura se desea consultar.

2.- Comprobar que existe el descriptor del archivo de datos indicado.

3.- Presentar a la vista del usuario la estructura solicitada.

4.- Regresar el control al sistema.

La figura II.2. muestra el diagrama de flujo de la función crear.

II.3. FUNCION RENOMBRAR

Sus actividades principales son:

1.- Solicitar nombre del archivo de datos.

2.- Comprobar que para el archivo de datos indicado existe su descriptor en ADA.HDR.

3.- Solicitar otro nombre de archivo de datos.

4.- Comprobar que no existe un descriptor para el archivo especificado en el inciso anterior.

5.- Obtener de ADA.HDR el descriptor del archivo solicitado en inciso 1.

6.- Sustituir en el descriptor el nombre del archivo por aquél solicitado en el inciso 3.

7.- Salvar el descriptor en ADA.HDR.

8.- Regresar el control al sistema.

La figura II.3. muestra el diagrama de flujo para esta función.

II.4. FUNCION COPIAR Y RENOMBRAR

Realiza lo siguiente:

Si se desea tener un descriptor de archivo similar a uno existente solo que relacionado con otro archivo de datos, esta función es útil y la secuencia de eventos que desarrolla son:

1.- Pedir nombre del archivo de datos cuyo descriptor es conocido.

2.- Solicitar nombre para el archivo de datos cuyo descriptor será igual al obtenido en el inciso anterior.

3.- Checar que no existe un descriptor para el archivo de datos del inciso anterior.

4.- Obtener el descriptor del archivo de datos del inciso 1.

5.- Reemplazar en el descriptor el campo que corresponde al nombre del archivo por aquél que fue pedido en el inciso 2.

6.- Buscar espacio para el nuevo descriptor y salvarlo en ADA.HDR.

7.- Actualizar encabezado del archivo ADA.HDR.

8.- Regresar el control al sistema.

La figura II.4. muestra el diagrama de flujo para esta función.

II.5. FUNCION ELIMINA

Anula el descriptor del archivo de datos especificado.
Su funcionamiento es el siguiente:

1.- Pide el nombre del archivo de datos cuyo descriptor se desea cancelar.

2.- Verifica que exista el descriptor para el archivo indicado.

3.- Marca el área que utiliza el descriptor como espacio disponible.

4.- Actualiza encabezado del archivo ADA.HDR.

5.- Regresa el control al sistema.

La figura II.5. muestra el diagrama de flujo para esta función.

II.6. FUNCION ELIMINAR

Esta rutina no solamente anula la estructura descriptiva de un archivo de datos, sino que también remueve físicamente el archivo de datos.

Actua de la siguiente manera:

1.- Solicita el archivo que se desea anular.

2.- Se localiza su descriptor en ADA.HDR.

3.- Se marca el descriptor a cancelar.

4.- Se borra el archivo de datos respectivo.

5.- Se actualiza la cabeza de lista del archivo ADA.HDR.

6.- Se regresa el control al sistema.

La figura II.6. muestra el diagrama de flujo para esta función.

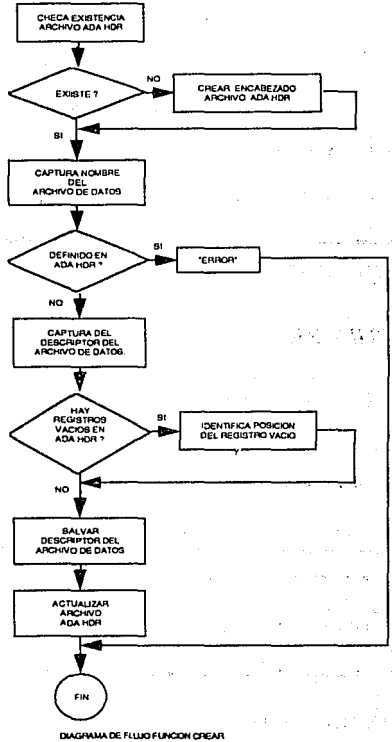


DIAGRAMA DE FLUJO FUNCION CREAR

Figura II.1.

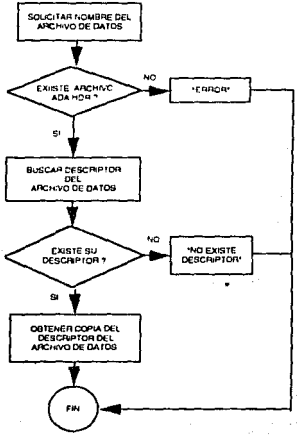


DIAGRAMA DE FLUJO FUNCION CONSULTAR

Figura II.2.

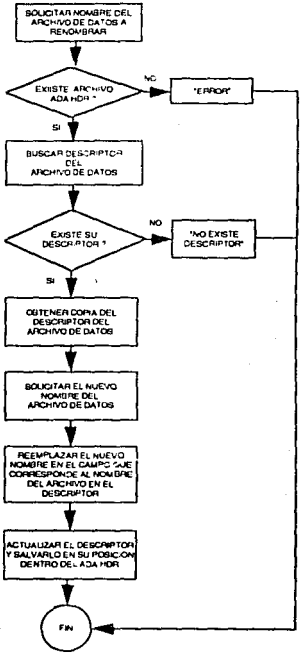


DIAGRAMA DE FLUJO FUNCION RENOMBRAR

Figura # 3

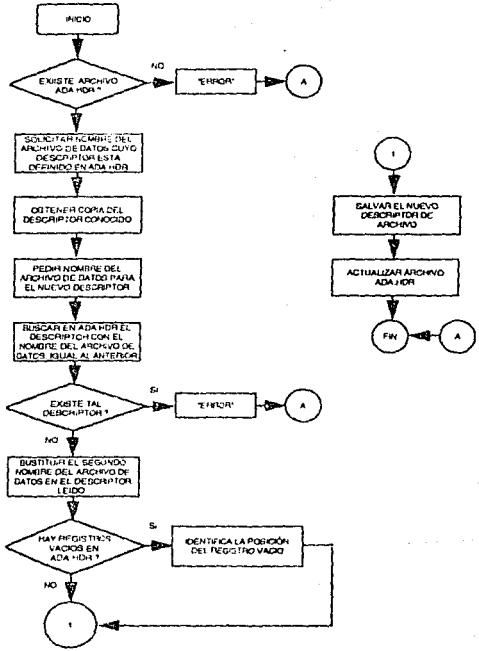


DIAGRAMA DE FLUJO FUNCION COPIAR Y RENOMBRAR

Figura 6.4

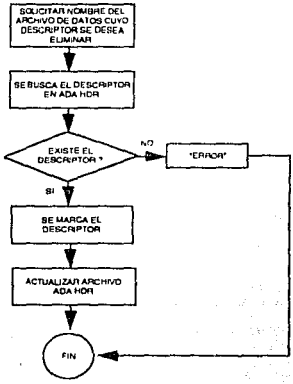


DIAGRAMA DE FLUJO FUNCION ELIMINA

Figura # 5

DIAGRAMA DE FLUJO FUNCION ELIMINAR

Figura 8.8.

CAPITULO III

OPERACIONES SOBRE LOS ARCHIVOS DE DATOS

Solo el ejemplo de los individuos grandes y puros puede llevarnos a pensamientos y acciones nobles.

III) OPERACIONES SOBRE ARCHIVOS DE DATOS

III.1. INTRODUCCION

Esta fase de desarrollo comprende todas aquellas operaciones que se realizan sobre los archivos de datos, como lo son :

- Agregar información
- Actualizaciones sobre la información existente.
- Borrar información.
- Consultas.
- Compactación de archivo.

Para ello se definieron dos estructuras de datos fundamentales, la primera de ellas utilizada para llevar el manejo de la información correspondiente a la descripción del archivo a utilizar (número de campos, longitud de registro, así como también el nombre, tipo y longitud de cada uno de los campos), esta estructura se genera a través del archivo descriptor de archivos (ADA.HDR). Por otro lado tenemos la segunda estructura de datos que se emplea para establecer el punto de enlace entre la información de la base de datos y el usuario, es decir, sobre esta estructura se llevan a cabo las consultas y actualizaciones de información de la base de datos.

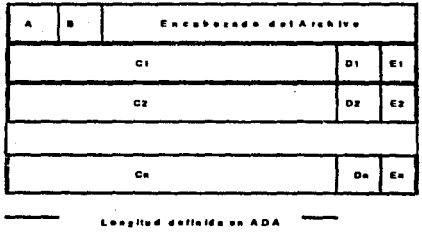
Una vez establecidas estas dos estructuras el usuario se encuentra en disposición de realizar las operaciones anteriormente señaladas, teniéndose una rutina principal que se encarga de identificar la operación a realizar y de llevarla a cabo. Esta rutina a su vez se auxilia de una serie de subrutinas para su operación. Además, durante el proceso de ejecución se lleva el estado de las

operaciones realizadas de tal forma que se mantiene al usuario informado del correcto desempeño de sus operaciones.

III.2. ESTRUCTURAS DE LOS ARCHIVOS DE DATOS

Los archivos de bases de datos tienen la estructura que a continuación se define:

ESTRUCTURA DE UN ARCHIVO DE DATOS.



D E S C R I P C I O N:

A1 **:**

Forma parte del encabezado del archivo teniendo una longitud de cuatro bytes y nos sirve para saber el número de registros grabados en el archivo en un instante determinado, este número se encuentra almacenado en forma de caracteres, teniéndose como máximo 65535 registros.

Este campo se actualiza con frecuencia ya sea al agregar nuevos registros o bien al borrar algunos.

B1 **:**

Apuntador al primer registro disponible, es decir en estos bytes se almacena el número del primer registro disponible (último registro borrado), inicialmente este campo es nulo.

C1..Cn **:**

Información de cada uno de los registros.

D1..Dn **:**

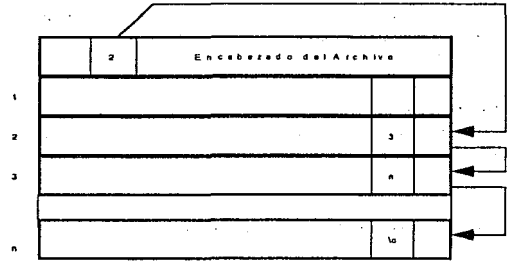
Apuntador al siguiente registro disponible.

E1..En **:**

Este byte se utiliza para marcar un registro para posteriomente ser borrado, la marca que se pone es un asterisco "*". Los registros que son marcados se borran en forma definitiva al compactar el archivo.

Como se puede observar en la estructura anteriormente definida la longitud total del registro es la definida en ADA más otros 3 bytes. Ahora bien, es importante señalar la forma en que se manejan los registros disponibles, para ello diremos que cada vez que se genera un registro disponible (borrado), el número de dicho registro se coloca en el apuntador del encabezado y el número de registro anterior contenido

en dicho encabezado es colocado en el apuntador hacia el siguiente disponible del registro borrado, con lo que se genera una lista ligada con apuntadores a registros disponibles (ver Figura).



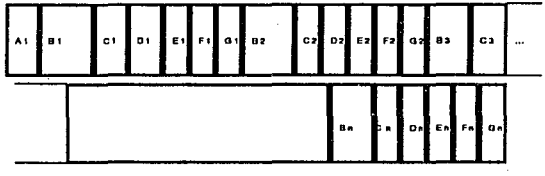
Manejo de Registros Disponibles

III.3. ESTRUCTURAS DE DATOS FUNDAMENTALES

A continuación se describen las estructuras de datos utilizadas en esta fase.

III.3.1.VECTOR DESCRIPTOR DE CAMPOS (VDC).

Este vector tiene el siguiente formato:



Donde:

AI **:**

Número de campos del registro (longitud de 2 bytes).

BI..Bn **:**

Nombre del campo (longitud de 5 bytes).

CI..Cn **:**

Tipo del campo (longitud de 1 byte).

DI..Dn **:**

Referencia a tabla (longitud de 1 byte). No utilizado

EI..En **:**

Longitud del campo (2 bytes).

F1..Fn :

Número de decimales (1 byte).

G1..Gn :

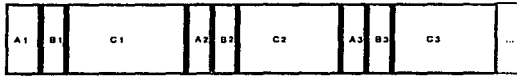
Posición del campo dentro del registro (4 bytes).

Este vector se genera a partir del archivo descriptor de archivos (ADA), es decir, una vez proporcionado el nombre del archivo con el cual se va a trabajar se procede a buscar la información correspondiente, realizándose para ello una búsqueda en ADA antes de iniciar cualquier operación sobre los archivos de datos. Además, posee un tamaño variable por lo que se utilizó un apuntador para su almacenamiento. Ahora bien, pueden ocurrir dos problemas al tratar de obtener esta información, la primera de ellas es que el archivo solicitado no se encuentre definido en ADA, careciéndose por lo tanto de una estructura definida no pudiéndose entonces iniciar el proceso de acceso a la información existente en el archivo de base de datos. Cuando ocurre esto el usuario es notificado del problema existente. Por otro lado, el archivo solicitado puede estar definido en ADA pero no ha sido creado en disco, es decir, no se ha definido un encabezado para su utilización, en este caso se procede a crear el archivo con su encabezado correspondiente.

El objetivo de este vector es mantener activa la información de todos los campos para que en cualquier momento se pueda hacer la manipulación de la información en la forma adecuada. Es de importancia señalar que este vector se define una sola vez, ya que definido se conserva hasta el final del proceso. Cualquier operación ya sea de consulta, actualización y borrado precisa de una consulta a este vector antes de proceder a realizar la tarea correspondiente. Generalmente las consultas consisten en saber la longitud del campo a afectar, la posición que ocupa en el registro ó bien si existe alguna referencia a tablas.

III.3.2. VECTOR DE REGISTROS DEL ARCHIVO DE DATOS (VRAD).

Este vector tiene el siguiente formato:



Descripción de campos:

Ai..An :

Este byte se utiliza para indicar el estado del registro, es decir, los cambios que ha sufrido, teniéndose los siguientes estados posibles:

" " : Indica un registro normal, es decir, que no ha sufrido cambio alguno.

"@" : Indica que el registro fue sometido a una actualización en cualquiera de sus campos.

"*" : Indica que el registro que se cargó al generar el VRAD está marcado como borrado.

"!" : Indica que se desea borrar el registro en cuestión. La diferencia entre este estado y el anterior es que el anteriomente mencionado involucra a un registro que ya está grabado en el archivo de datos, mientras que "!" se refiere a un registro que está próximo a borrarse.

B1..Bn :

Este campo se utiliza para almacenar el número de registro físico del registro correspondiente.

C1..Cn :

En este campo se almacena la información del registro correspondiente.

FUNCION:

El VRAD es un vector que se crea dinámicamente y que se utiliza para el manejo de un grupo de registros pertenecientes a un archivo de bases de datos determinado, mediante este vector se llevan a cabo las operaciones de obtención ó actualización de la información de cualquier campo contenido en un registro determinado.

El rango entre el cual se toma este grupo de registros se establece tomando como base el número de registro solicitado, de tal forma que usted quede colocado en la mitad del rango teniendo con esto la posibilidad de manejar la información de registros anteriores o posteriores al solicitado. Otras consideraciones para el establecimiento del rango es la cantidad de memoria disponible y el tamaño del archivo de tal forma que se puedan cargar tantos registros como sea posible, claro está, sin agotar la memoria disponible.

El VRAD se actualiza cada vez que se solicita un número de registro fuera del rango establecido en ese momento, cuando esto sucede se graban todas las actualizaciones realizadas y una vez realizado esto se procede a generar de nueva cuenta a dicho vector.

Las operaciones que se efectúan sobre este vector comprenden:

Inserción o actualización de información. Estas operaciones consisten en escribir la información en el campo y registro correspondientes.

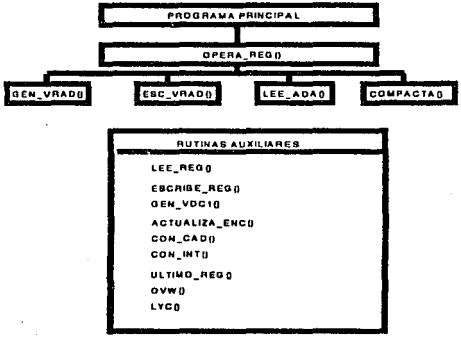
Borrado de información. El borrado consiste únicamente en marcar el registro y definirlo como disponible. Posteriormente, si es que así se desea se podrán borrar físicamente del archivo.

Consulta de información. Se extrae la información del campo y registro correspondientes.

III.4. RUTINAS PRINCIPALES Y AUXILIARES.

La coordinación y desarrollo de todas las operaciones que se efectúan sobre los archivos de datos se lleva a cabo mediante una serie de rutinas, algunas de las cuales son de gran importancia y las demás nos proporcionan la ayuda en determinados procesos.

A continuación se describirán cada una de estas funciones en orden de importancia.



III.4.1.RUTINAS PRINCIPALES.

RUTINA :

```
OPERA_REG(CADENA, SELECTOR, APT, CAMPO, REG_SOL)
  char *cadena, *apt, selector;
  int  campo,reg_sol;
```

PARAMETROS:

CADENA :

Es un apuntador a la cadena que se proporciona como entrada a la rutina opera_reg, puede corresponder a la información a grabar o bien al nombre del archivo a utilizar.

APT:

Apuntador a la cadena en la cual se almacena la información resultante, como puede ser : la información del campo solicitado, el vector con información de los campos, o bien el número de registro a agregar.

SELECTOR :

Este parámetro nos sirve para seleccionar la operación a realizar.Las posibles opciones son:

0 : Obtener la información correspondiente a los - campos (nombre, tipo, longitud y posición), es decir,el VDC.

1: Agregar información al final del archivo.
2: Editar información de un registro, al realizar esto nos posicionamos en el primer campo del primer registro.
3: Consultar información de un campo en un registro determinado.
4: Escritura de información en un campo.
5: Borrar un registro.
6: Finalizar la utilización de un archivo, realizándose la compactación del mismo.
7: Obtención del número de registros grabados - en el archivo incluyendo aquellos registros que están borrados.

CAMPO:

Número del campo a utilizar.

REG_SOL:

Número de registro a utilizar.

FUNCION

Esta rutina es la principal en el proceso de manejo de archivos ya que se encarga de organizar y efectuar las diferentes operaciones, para ello se auxilia de algunas otras rutinas. La correcta ejecución de toda operación es controlada a través de un estado que se lleva a lo largo del proceso. Estos estados se definen como a continuación se especifica:

Código	Estado
0	Ejecución correcta
1	No existe ADA
2	Archivo no existente en ADA
3	Archivo de datos no creado
4	Campo inexistente
5	Registro borrado (consulta) ó registro que ya fue borrado (borrado).
.	.
.	.
.	.

RUTINA

```
GEN_VRAD (VRAD,  REG_INIC,   REG_FIN,   NUMREG_ADD,   LONREG,
          SOLIC_DISP,NOMB, REG_DEL, REG_SOL);
  CHAR *VRAD;
  INT  *REG_INIC,*REG_FIN,NUMREG_ADD,LONREG,*SOLIC_DISP;
  CHAR NOMB[25]; INT  REG_DEL, REG_SOL;
```

PARAMETROS

VRAD :

Apuntador en donde se almacena el vector de registros del archivo de datos (VRAD).

REG_INIC :

Número de registro inicial del rango.

REG_FIN :

Número de registro final del rango.

NUMREG_ADD:

Número de registros del archivo de datos sin contar aquellos que fueron borrados.

LONREG:

Longitud de registro.

SOLIC_DISP:

En esta variable se lleva el número de registros disponibles, es decir, sin utilizar dentro del rango establecido.

NOMB:

Nombre del archivo de datos.

REG_DEL:

Número de registros borrados.

REG_SOL:

Número de registro solicitado.

FUNCION

Esta rutina se encarga de leer un grupo de registros almacenados en disco y almacenarlos en el vector de registros del archivo de datos (VRAD). Para ello primeramente establece un rango (registro inicial y final) de acuerdo al registro solicitado de tal forma que usted quede colocado a la mitad del rango. Una vez establecido se procede a la lectura de disco y almacenamiento de información.

RUTINA

```
ESC_VRAD(VRAD,  LONREG,   REG_GRAB,   SOLIC_DISP,   DISP_ADD,
         NUMREG_ADD,NOMB, REG_DEL, REG_REC, REG_INIC);
    CHAR  *VRAD;
    INT   LONREG, *REG_GRAB,SOLIC_DISP, *DISP_ADD,NUMREG_ADD;
    CHAR  NOMB[25]; *REG_DEL, *REG_REC, REG_INIC;
```

PARAMETROS

VRAD:

Apuntador en donde se almacena la información del grupo de registros.

LONREG :

Longitud de registro.

REG_GRAB:

Número de registros recuperados.

SOLIC_DISP:

Número de registros disponibles en el rango (no tienen información).

DISP_ADD:

Apuntador al primer registro disponible que se almacena en el encabezado.

NUMREG_ADD:

Número de registros del archivo de datos (sin marcar).

NOMB:

Nombre del archivo de datos a utilizar.

REG_DEL:

Número de registros borrados.

REG_REC:

Número de registros recuperados.

REG_INIC:

Número del registro inicial del rango.

FUNCION

Esta rutina es utilizada para efectuar todas las modificaciones registradas en el VRAD, en las que se encuentran la actualización y el borrado de información perteneciente al grupo de registros definido en

ese momento. Esta rutina se utiliza cada vez que se tiene que generar otro VRAD o bien cuando se termina de trabajar con el archivo de datos. Trabaja únicamente con aquellos registros que sufrieron alguna modificación, identificándolos por la marca que se encuentra colocada en ellos y que indica si el registro fue modificado ("@") o bien borrado ("!").

RUTINA

```
LEE_ADA(VDC, NOM_ARC, LONREG, NUMCAMP);
    CHAR    *VDC, NOM_ARC[25];
    INT     LONREG, NUMCAMP;
```

PARAMETROS

VDC :

Apuntador que almacena la información de todos los campos que forman parte de la estructura del archivo de datos (VDC).

NOM_ARC:

Nombre del archivo de datos.

LONREG:

Longitud de registro.

NUMCAMP:

Número de campos que forman el registro del archivo de datos.

FUNCION

Rutina utilizada para obtener la información correspondiente al archivo especificado, misma que se encuentra almacenada en el archivo descriptor de archivos (ADA). Si ésta existe se genera el vector

descriptor de campos (VDC), en caso contrario, se indica un estado de error.

Por otra parte se obtiene la longitud de registro y el número de campos.

RUTINA

```
COMPACTA(NOMB,      NUMREG_ADD,      DISP_ADD,      REG_DEL,      LONREG);
CHAR  NOMB[25]; UNSIGNED INT   NUMREG_ADD, DISP_ADD, REG_DEL, LONREG;
```

PARAMETROS

NOMB **:**

Nombre del archivo a utilizar.

NUMREG_ADD:

Número de registros del archivo de datos.

DISP_ADD:

Apuntador al primer registro disponible.

REG_DEL:

Número de registros borrados.

LONREG:

Longitud de registro.

FUNCION

La compactación consiste en la eliminación, mediante su reutilización de todos aquellos registros que fueron borrados de tal forma que el archivo contiene finalmente registros con información

útil. Además, permite quitar la marca ("*") en aquellos registros que fueron recuperados. El proceso utilizado es el siguiente: se van tomando los últimos registros del archivo de datos y se reubican en los registros disponibles de tal forma que los disponibles se toman del encabezado, actualizándose con ello el nuevo apuntador del encabezado. Ahora bien, como tenemos la posibilidad de recuperar un registro borrado, este tipo de registros conservan el apuntador al siguiente disponible por tanto al realizar la compactación se toma únicamente el apuntador y posteriomente se blanquea este valor en el registro dejando intacta la información contenida en este campo.

Este proceso se desarrolla hasta agotar la lista de disponibles existentes.

III.4.2. RUTINAS AUXILIARES.

RUTINA

```
LEE_REG(FP, REG, POS, TAM, NUMREGS);
   FILE    *FP;
   CHAR    *REG;
   INT     POS, TAM, NUMREGS;
```

PARAMETROS

FP:

Variable para utilización del archivo.

REG:

Apuntador en el cual se almacena la información leída del archivo en disco.

POS :

Posición dentro del archivo a partir de la cual se leerá la información.

TAM:

Longitud de la información que se va a leer.

NUMREGS:

Número de elementos con el tamaño anterior que se van a leer.

FUNCION

Esta función se utiliza para leer un conjunto de registros del archivo de datos especificado. Se leen "NUMREGS" elementos de longitud "TAM" a partir de la posición "POS" del archivo "FP"

RUTINA

```
ESCRIBE_REG(FP, REG, POS, TAM, NUMREGS);
   FILE   *FP;
   CHAR   *REG;
   INT    POS, TAM, NUMREGS;
```

PARAMETROS

FP:

Variable utilizada para el manejo del archivo.

REG:

Apuntador en el cual se almacena la información que se desea escribir en el archivo.

POS:

Posición dentro del archivo a partir de la cual se grabará la información.

TAM:

Longitud de la información que se grabará en el archivo.

NUMREGS:

Número de elementos con el tamaño anteriormente especificado que se desean escribir en disco.

FUNCION

Esta rutina se utiliza para grabar uno o más registros en el archivo de datos especificado.

RUTINA

```
GEN_VDC1( VDC, VDC1, NUMCAMP);
        CHAR    *VDC, *VDC1;
        INT     NUMCAMP;
```

PARAMETROS

VDC :

Apuntador en el cual se encuentra almacenado el vector descriptor de campos (VDC)

VDC1 :

Apuntador en el cual se almacena la información que requiere el programa principal (número de campos, nombre, tipo, decimales y longitud de cada campo).

NUMCAMP:

Número de campos.

FUNCION

Esta rutina genera la información de los campos necesaria para la captura de información en pantalla consistente en el número de campos, su nombre, tipo, decimales y longitud.

RUTINA

```
ACTUALIZA_ENC(NUMREG, DISPON, LONREG, NOMB);
    UNSIGNED INT NUMREG, DISPON;
    INT LONREG;
    CHAR NOMB[25];
```

PARAMETROS

NUMREG :

Número actualizado de registros del archivo de datos.

DISPON :

Apuntador al primer registro disponible actualizado.

LONREG:

Longitud de registro del archivo de datos.

NOMB:

Nombre del archivo de datos a utilizar.

FUNCION

Mediante esta rutina se lleva a efecto la actualización del encabezado del archivo de datos requiriendo para ello del número de registros y el apuntador al primer disponible. Una vez recibidos estos parámetros se convierten de números a cadenas y se concatenan. Finalmente se graba el nuevo encabezado en el archivo.

RUTINA

```
CON_CAD(S1, NUMERO);
    UNSIGNED CHAR *S1;
    UNSIGNED INT  NUMERO;
```

PARAMETROS

S1:

Apuntador que regresa una cadena compuesta de dos caracteres generados por la conversión del número a cadena.

NUMERO:

Número que se convertirá a cadena.

FUNCION

Rutina utilizada para convertir un número entero menor o igual a 65535 en una cadena formada por dos caracteres.

RUTINA

```
CON_INT(S1, NUMERO);
   UNSIGNED CHAR S1[3];
   UNSIGNED INT  *NUMERO;
```

PARAMETROS

S1:

Cadena recibida para transformarla en un número.

NUMERO:

Variable en la que se regresa el valor resultante de la conversión de la cadena anterior.

FUNCION

Esta rutina se utiliza para convertir una cadena formada por dos caracteres en un número menor o igual que 65535.

RUTINA

```
ULTIMO_REG(NOMB, LONREG);
   CHAR NOMB[25];
   INT LONREG;
```

PARAMETROS

NOMB :

Nombre del archivo de datos a utilizar.

LONREG:

Longitud de registro.

FUNCION

Esta rutina se utiliza para obtener el último número de registro grabado en el archivo.

CAPITULO IV

ORDENAMIENTO DE LOS ARCHIVOS DE BASES DE DATOS

Quien tiene salud tiene esperanza;
y quien tiene esperanza lo tiene todo.

IV. ORDENAMIENTO DE LOS ARCHIVOS DE BASES DE DATOS

Uno de los atractivos de un sistema manejador de archivos de una base de datos es su habilidad para ordenar registros de información de manera que podamos disponer la información en el orden que deseemos: alfabético, cronológico, numérico ó por una combinación de ellos.

Por lo anterior el sistema AVE incluye un método para llevar a cabo el proceso de ordenamiento de archivos. Los métodos utilizados son, por un lado, el de indexación, el cual es contemplado en el capítulo V (no implementado en este sistema); por otro, contemplamos la alternativa para ordenar y que concierne a esta sección pertenece a aquellas consideradas como clásicas donde todo el proceso esencial se lleva a cabo en memoria principal utilizando algún algoritmo bastante conocido, en este caso se utilizó el método de Quick Sort.

En las siguientes líneas se expresan las ideas que soportan al algoritmo desarrollado.

Una vez que se ha recibido el nombre del archivo de datos a ordenar y comprobar que la petición de ordenamiento tiene la sintaxis correcta, se procede a investigar que los campos por los cuales se desea llevar a cabo la ordenación están definidos en el descriptor del archivo; satisfecho lo anterior se determina la cantidad de memoria RAM contigua disponible a fin de poder hacer una evaluación de si el archivo de datos cabe completo en RAM o no.

Si el caso es que puede cargarse en su totalidad se procede a su lectura y carga y el ordenamiento se lleva a cabo sin contratiempo. Por

el contrario, esto es, si no es posible cargar el archivo de datos en una sola operación de lectura por falta de espacio, se continúa con la determinación de la cantidad máxima de registros que si pueden residir en RAM, así también se hace el cálculo del número de bloques en que puede ser dividido el archivo de datos.

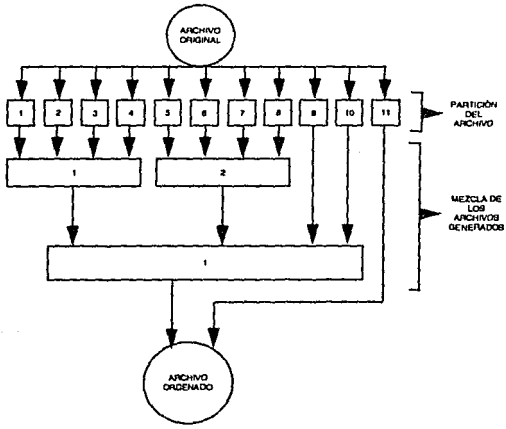
Por un bloque vamos a entender la cantidad máxima de registros que puedan ser cargados en RAM en una sola operación de lectura.

Con la información anterior se procede a cargar y ordenar uno a uno los bloques de registros en que se ha subdividido el archivo de datos original. En este punto se genera un archivo de datos temporal por cada bloque procesado.

Entonces el siguiente paso consiste en mezclar en grupos de cuatro los archivos generados y como resultado, se genera cada vez un nuevo archivo, producto de la mezcla.

El proceso de mezclar archivos se prosigue hasta que en la última etapa se combinan cuatro o un número menor de archivos y se obtiene el archivo final ya ordenado.

Gráficamente, el proceso descrito se ilustra en la gráfica IV.1.



PROCESO DE ORDENADO Y MEZCLADO DE UN ARCHIVO

Figura IV.1

El diagrama de flujo de la función de ordenar se presenta en la gráfica IV.2.

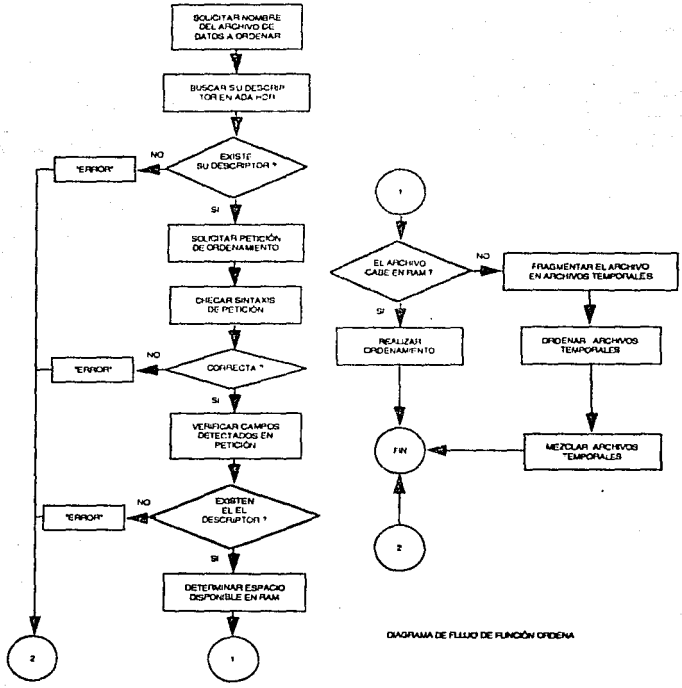


DIAGRAMA DE FLUJO DE FUNCIÓN ORDENA

Figura IV.2

CAPITULO V

INDEXACION

Un hombre sabio creará mas oprtunidades de las que halle.

V) INDEXACION

V.1. INTRODUCCION.

El problema de acceso a los datos es fundamental en la utilización de la computadora. Aún cuando el procesamiento automático de datos es rápido en comparación con los métodos manuales, el tiempo requerido para almacenar y recuperar información lo más pronto posible, llega a ser significante cuando se involucran grandes cantidades de datos.

El establecimiento de índices en una base de datos permite entre otras cosas localizar cualquier registro en el archivo en un tiempo muy pequeño. Otra conveniencia del índice es que permite conservar los registros del archivo ordenados sea alfabética, cronológica o numéricamente.

Entre los diferentes métodos desarrollados para el acceso rápido a los datos se eligió el basado en el principio de los árboles B (nombrados así por uno de sus creadores R. Bayer). En un árbol B una unidad de datos es accesada mediante la utilización de una llave. Cualquier llave está relacionada con una y sólo una unidad de datos en un archivo de datos. En el árbol B, las llaves están organizadas de tal forma que la búsqueda de una llave particular requiere de muy pocos accesos al almacenamiento en disco.

La idea de este proceso fue la de tener la posibilidad de mantener varios archivos de datos y accesar cada uno de estos archivos con diferentes llaves. Además de que los diferentes archivos de datos pueden ser accesados utilizando la misma llave.

En la mayoría de los casos, las unidades de datos a ser almacenadas o recuperadas ocupan mucho más espacio que sus llaves.

Esto retarda el proceso de búsqueda y ésta es una de las razones por la cual la relación entre llaves y unidades de datos a menudo se almacenan en archivos especiales separados. Al archivo que contiene las llaves lo denominamos Archivo Indexado y a los archivos que contienen únicamente unidades de datos al ser almacenados ó recuperados los denominamos Archivo de Datos.

Existen varias acciones posibles involucradas en el manejo de los archivos de índices organizados como árboles B, entre las que se encuentran: Búsqueda por una llave en particular; Inserción ó borrado de una llave junto con su elemento de datos asociado.

V.2. ESTRUCTURA DE LOS ARBOLES

V.2.1. Llaves

El usuario maneja los archivos de datos con la ayuda de referencias de datos almacenadas en los Archivos de índices. Estas referencias son almacenadas y recuperadas por medio de llaves. Las llaves son cadenas de caracteres especificadas por el usuario y que tienen una longitud de hasta 255 caracteres. El principio de los árboles B asume que no es posible tener dos referencias de datos con la misma llave.

V.2.2. Elementos (ITEMS)

La conexión entre una llave y su referencia hacia un archivo de datos es formalizado mediante un registro llamado **ELEMENTO**.

Además de estos dos componentes, el elemento maneja una referencia interna en el Archivo de Indices (Referencia de Página), la cual es utilizada en la construcción de la estructura de árbol

V.2.3. Páginas

Las unidades de un Archivo de índices son llamadas PAGINAS Cada página contiene un número par de elementos. La primera página es denominada página raíz. Todas las demas páginas contienen al menos la mitad de sus elementos con sus respectivas llaves y sus referencias a páginas y datos. Este es el porqué el orden (n) del árbol B es definido como la mitad del número máximo de elementos por página. Además de estos elementos, la página contiene una referencia extra de página la cual no está conectada a ningún elemento.

V.2.4. Páginas dentro de Arboles.

Como hemos visto, cada llave en un elemento de página maneja una posible referencia hacia otra página de la estructura de datos. En un árbol B, la página hacia la cual la referencia apunta contiene llaves adicionales que son mayores. Así pues, los saltos consecutivos de un elemento de página hacia la página de su referencia va creando una trayectoria que finaliza en la página terminal la cual no posee referencias hacia otras páginas. Estas páginas son llamadas páginas hojas.

También es posible moverse de un página a otra de tal forma que cualquier llave en la segunda página menor a cualquier llave de la primera página. Esto es realizado mediante la utilización de una referencia extra de página contenida en todas las páginas.

Todas las trayectorias en el árbol B empiezan en la página raíz, y cada salto puede ser ya sea a llaves menores ó bien a llaves mayores.

El ordenamiento de las llaves en un árbol B tiene como

consecuencia que cada página en el árbol puede ser alcanzada mediante una y sólo una trayectoria. El número de saltos en esta trayectoria se referencia como el nivel de la página.

V.2.5. Búsqueda de Referencias de Datos.

El propósito de un árbol B es proveer una referencia hacia un archivo de datos mediante la especificación de una llave. La página con la referencia de dato es encontrada siguiendo su trayectoria iniciando en la página raíz. Este viaje es guiado por la comparación con las llaves encontradas en las páginas a lo largo del recorrido y que nos proporcionan la información exacta del siguiente salto. Si todas las llaves de una página son mayores a la llave buscada, la siguiente página a investigar es la referenciada por la referencia de página extra. De otra forma, la referencia hacia la siguiente página se encuentra en el elemento de datos con la llave mayor, la cual es menor a la llave buscada. La búsqueda continúa hasta que la llave es encontrada ó bien hasta que se alcanza la página hoja sin resultado alguno.

V.2.6. Propiedades Formales de los Arboles.

La estructura de los árboles B está sujeta a varias restricciones para asegurar que la eficiencia de búsqueda bajo cualquier circunstancia de inserción ó de borrado de llaves quede intacta. Las restricciones, algunas de las cuales ya han sido mencionadas, son las siguientes:

- Todas las páginas contienen al menos 2 elementos (llaves).
- Con excepción de la página raíz, todas las páginas contienen al menos n elementos (llaves).
- Toda página es ya sea una página hoja sin ninguna referencia de página activa ó bien posee m+1 referencias de páginas activas (donde m es el número de llaves de la página (n<=m<=2n)).
- Todas las páginas hojas están al mismo nivel (denominada altura del árbol).

Las ventajas generadas al asumir las reglas anteriores son varias. Por una parte aseguran que todas las partes del árbol B poseen una densidad de información mínima. Al mismo tiempo, la mayoría de las llaves tienen la misma longitud de trayectoria, esto es, la altura actual del árbol es la misma.

Cada vez que la altura del árbol B se incrementa en uno debido a la inserción de nuevas llaves, se incrementa el espacio para las llaves adicionales en un factor entre (n+1) y (2n+1) comparado con el nivel previo.

V.2.7. Inserción de Llaves.

Cuando se desea insertar una llave en el árbol B, primeramente se realiza un chequeo para determinar si la llave en cuestión ya existe o no en el árbol. Si la llave es nueva, la búsqueda termina en la página hoja con un resultado negativo. Siendo esta misma hoja el lugar apropiado para la inserción de la nueva llave. El único problema que

puede ocurrir es cuando la página hoja está llena (Esto es, ya contiene 2n elementos).

Este problema se resuelve mediante la redistribución de las (2n+1) llaves en dos páginas una de las cuales es nueva. Las n llaves más grandes se mueven a la nueva página hoja y la llave número (n+1) se mueve a la página ancestral a la página hoja, donde es asociada mediante una referencia hacia la nueva hoja. Esto preserva el orden de las llaves afectadas por la operación.

· Si la página ancestral está llena, se debe realizar una redistribución hacia arriba en cualquier hoja. Cuando esto ocurre la división de la raíz resulta en la creación de una nueva raíz y el árbol B crece en su altura.

V.2.8. Borrado de Llaves.

Cuando se solicita el borrado de una llave existente ésta se localiza en el árbol . Si se encuentra ubicada en una página hoja su borrado es simple, pero si se encuentra en alguna otra parte del árbol el proceso de borrado es más complicado.

La referencia de página vacante debe ser conectada a otra llave del árbol sin destruir el orden. Esta llave debe ser de una hoja misma que puede ser utilizada sin afectar el ordenamiento de las llaves.

Ahora bien, al mover la llave de la página hoja puede desbalancear el árbol ya que se disminuye el número de elementos, para ello es necesario realizar una redistribución hacia abajo.

V.2.9. Rutinas.

Las principales componentes de esta fase de indexación comprenden el establecimiento de las siguientes rutinas:

ACCESO:

Contiene las rutinas básicas para el mantenimiento y establecimiento de los archivos de índices.

CONSULTA_LLAVE:

Contiene las rutinas de búsqueda para realizar las operaciones de manejo de las llaves.

AGREGA_LLAVE:

Contiene las rutinas para inserción de llaves en los archivos de índice.

BORRA_LLAVE:

Contiene las rutinas utilizadas para el borrado de llaves.

CAPITULO VI

QUERY

A los ojos del hombre de imaginación,
la naturaleza es la imaginación misma.

VI. QUERY

El filtrado de información de grandes cantidades almacenadas de datos representa una característica muy valiosa y necesaria en el manejo de bases de datos. Así el usuario que desea realizar una consulta la puede hacer por métodos automatizados sobre toda la información, obteniendo como resultado final únicamente los datos que cumplan ciertas reglas o características.

El caso de AVE no se aparta de este concepto, se ha implementado una forma en la que el usuario final podrá poner filtros para manejar su información. Esta forma de filtrado se ha llamado "Query", mediante la cual el usuario construye una expresión en la que indica exactamente que información de toda la base de datos es la que le interesa.

En AVE, el proceso de Query se utiliza en una gran cantidad de procesos, Selección de Información para Edición, para Borrado, así como Exportación de información hacia otros formatos. El sistema AVE asumirá que se aplicará el proceso cuando identifique una expresión correcta de query, en caso contrario (cuando la expresión esté en blanco), el proceso de query no se aplicará en ninguna operación.

En realidad, la expresión de query antes de ser ejecutada es analizada en un principio, para determinar si están correctas todas las variables implicadas; en caso de tener éxito se revisa su estructura y si está correcta se procederá a ejecutar el contenido de la expresión para obtener finalmente un "si cumple" o un "no cumple".

Cabe mencionar que toda expresión de query en blanco genera un "si cumple" para cualquier conjunto de información presente. Un "no cumple" solo será arrojado si es que la expresión query es correcta y el bloque

de datos presente no cumple con las características deseadas y anotadas en la expresión de query.

A partir de este momento para formalizar la terminología utilizada en este trabajo, al proceso de revisión para las expresiones query se le denominará por "compilación para las expresiones query".

El proceso de compilación para expresiones query se encuentra dividido en tres partes :

SCANNER :

El método para la realización del scanner fue implementada mediante dos partes principales :

- Chequeo de sintaxis de la expresión query.
- Generación de las tablas de Campos, Enteros, Cadenas y Relaciones.

PARSER

En esta parte de la compilación se lleva a cabo el proceso en el cual se genera una tabla en la que se indica la secuencia a seguir para reducir la expresión de query dada.

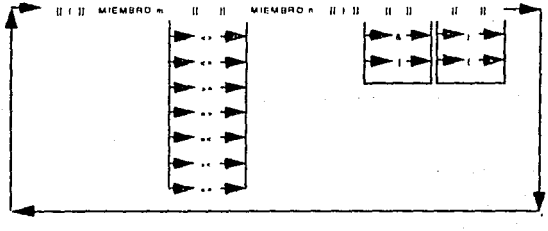
EVALUACION

Es aquí donde se lleva a cabo el contacto con la base de datos, y realizando las comparaciones indicadas en el vector de reducción arrojado por el proceso de parser.

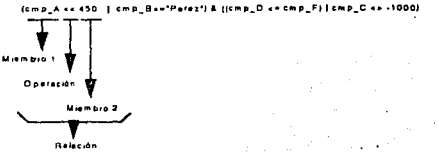
VI.1. SCANNER

Regularmente el proceso de scanner hablando en términos de lenguajes formales, realiza una comparación exhaustiva entre el texto dado y las palabras reservadas para el lenguaje predefinido. Es aquí donde se generan las tablas de símbolos, de enteros, de caracteres, etc. . Si hablamos del sistema AVE, podremos decir que este se ajusta completamente al proceso de scanner de la mayoría de los sistemas, con la excepción de que además se genera una tabla extra llamada "*Tabla de Relaciones*", en la cual se almacena información relevante para cada una de las relaciones dentro de la expresión.

La función scanner trabaja con la estructura de la Base de Datos en cuestión y la expresión dada, la cual está formada por los elementos mostrados en el siguiente diagrama :



ejemplo :



es decir, cada relación se encuentra formada por dos miembros y una operación booleana.

Dentro del chequeo sintáctico de la expresión en AVE se generan cuatro tablas :

- Tabla de Campos (TC, tabla número 1)
- Tabla de Enteros (TE, tabla número 2)
- Tabla de Cadenas (TS, tabla número 3)
- Tabla de Relaciones (TABL-REL),

las cuales tienen la siguiente estructura :

Tabla de Campos (Número 1)

# Renglón :	Indica un número secuencial de renglón.
Registro :	Número que ocupan los campos dentro de la definición de
la	base de datos.

Tabla de Enteros (Número 2)

# Renglón :	Misma función que la Tabla de Campos
Número :	Valores numéricos, enteros o reales indicados en la expresión.

Tabla de Cadenas (Número 3)

# Renglón :	Misma función que la Tabla de Campos.
Cadena :	Cadenas encontradas dentro de la expresión.

Tabla de Relaciones (Tabla_rel)

# Relación:	Contador secuencial del número de relación
Tabla A :	Número de Tabla a la cual se hace referencia del primer miembro de una relación.
Renglón A :	Número de Renglón al cual se hace referencia del número de Tabla indicada (Tabla A), del primer miembro de una relación.
Relación :	Indica la operación con que se relacionan los miembros de una relación. (==, <>, <=, etc).
Tabla B :	Número de Tabla a la cual se hace referencia del segundo miembro de una relación.
Renglón B :	Número de Renglón al cual se hace referencia del número de Tabla indicada (Tabla B), del segundo miembro de una relación.

Cada una de las primeras tres tablas contienen información obtenida de la estructura de la base de datos, y de la expresión.
Por ejemplo :

Dada la expresión :

```
(cmp_A << 450 | cmp_B=="Perez") & ((cmp_D <= cmp_F) | cmp_C <> -1000)
```

y teniendo una Base de Datos definida como :

Nombre del Campo	Tipo	Longitud	Posición en el Registro
cmp_A	Numérico	3	1
cmp_B	Alfanum.	10	2
cmp_C	Numérico	5	3
cmp_D	Fecha	8	4
cmp_E	Alfanum.	10	5
cmp_F	Fecha	8	6

El proceso de scanner arroja las siguientes tablas :

Tabla de Campos = TC = Tabla número 1

No. Renglón	Posición en el Registro
0	1
1	2
2	4
3	6
4	3

Tabla de Enteros = TE = Tabla número 2

No. Renglón	*Número*
0	*450.0*
1	*-1000.0*

Tabla de Strings = TS = Tabla número 3

No. Renglón	Cadena
0	Perez

Tabla de Relaciones = ABL-REL

# Relación	Tabla A	Renglón A	Relación	Tabla B	Renglón B
1	1	0	5 - <<	2	0
2	1	1	1 - ==	3	0
3	1	2	3 - <=	1	3
4	1	4	2 - <>	2	1

Cada renglón de esta tabla indica una relación y cada relación está formada por dos miembros y un signo de relación. Esto es, si tomáramos la relación número 2 de la tabla de relaciones se interpretaría de la siguiente forma :

Relación = 2 Relación número dos dentro de la expresión.

Tabla A = 1	Nos menciona que el valor será tomado de la tabla de Número = 1 (Tabla de Campos).
Renglón A = 1	Renglón Número 1 de la Tabla de Campos. Si se ve este valor se notará que estos datos corresponden al campo con posición en el registro = 2 y observando a la estructura de la base de datos esta posición corresponde al campo con nombre = cmp_B.
Relación = 1	Dentro de una tabla de símbolos el número 1 es asignado al símbolo "=="
Tabla B = 3	Se refiere a la Tabla Número 3 (Tabla de Cadenas)
Renglón B = 0	Renglón número 0 de la Tabla número 3, el cual corresponde al dato "Perez".

Entonces, podemos hablar que los datos de la tabla de relaciones :

2 1 1 3 0 corresponden a la relación cmp_B == "Perez"

Además de arrojar estas tablas, el proceso de scanner reduce la expresión a su mínima representación, es decir, únicamente se representa a las relaciones mediante un número el cual tiene su correspondencia y definición en la tabla de relaciones, quedando de la siguiente forma :

(1 | 2) & ((3) | 4)

Esta expresión recibe el nombre de *"Expresión Relacional"*.

VI.2. PARSER

El método seguido para la construcción del parser es el llamado "Parser Descendente Recursivo" el cual basa su funcionamiento en el llamado recursivo de funciones bajo ciertas reglas de producción previamente definidas.

La función PARSER tiene como principal objetivo indicar la secuencia adecuada para reducir la expresión mínima arrojada por el scanner. Dentro del parser se tienen algoritmos que indican el orden de prioridad para reducir una expresión tomando como variables los operandos y relaciones.

Dentro del sistema el parser genera una tabla de nombre "V_RED" (Vector de Reducción), con la siguiente estructura :

Estructura de la matriz V_RED :

1er Miembro :	Indica el número de casilla de algún vector en donde se almacenará el valor booleano del primer miembro de la relación.
Operación :	Operación booleana a ejecutarse entre los los miembros de una relación, (AND, OR).
2o Miembro :	Número de casilla de algún vector en donde se almacenará el valor booleano del segundo miembro de la relación.
Resultado :	Indica el Número de casilla de algún vector en donde se almacenará el resultado de aplicar la operación booleana indicada entre los dos miembros indicados.

Continuando con el ejemplo, si partimos de la expresión relacional arrojada por el scanner :

(1 | 2) & ((3) | 4)

entonces el parser nos arrojaría la matriz V_RED con los siguientes datos:

1er. Miembro en Casilla	Operación en Cuestión	2o. Miembro en Casilla	Resultado en Casilla
1	|	2	5
3	|	4	6
5	&	6	7

VI.3. EVALUACION

En este paso, es cuando se tiene contacto directo con los campos, registros y Base de datos, ya que se realizarán las comparaciones necesarias para determinar si el registro actual de la base de datos presente cumple o no las condiciones indicadas en cada miembro de la expresión (función EVALUA) y aún entre miembros (función EVALUA2).

Continuando con el ejemplo, esta tabla (V_RED) es utilizada por la función "EVALUA", la cual analizará los miembros existentes e indicados en cada renglón, el valor booleano del resultado de cada miembro se almacenará en un vector de resultados llamado "V_RES" en la casilla indicada por la tabla V_RED.

Si se observa la matriz V_RED, contiene un número mayor de miembros que en la expresión. La función EVALUA tiene la capacidad para determinar que miembros serán adquiridos de la expresión inicial y discriminar los que no estén.

Entonces, continuando con el ejemplo, después de la función EVALUA, el vector de resultados (V_RES) contendrá los siguientes valores :

V_RES

No. Renglón	Valor
1	1
2	0
3	0
4	0

esto es, si analizamos el primer miembro del vector encontraremos que el valor del miembro *(cmp_A << 450)* número 1 de la expresión inicial resultó tener un valor verdadero, es decir, la función *EVALUA* tomó el valor del campo llamado *cmp_A* del registro actual y encontró que este tiene un

valor menor que *450*, entonces almacena un valor cierto en el vector *V_RES* en la posición indicada en la matriz *V_RED*. El procedimiento continúa de igual forma con los siguientes renglones del vector *V_RES*.

Hasta este momento se ha determinado solo el valor booleano de cada miembro representado en la relación, pero no se ha obtenido un valor definitivo de la expresión, ya que para esto se requiere de realizar las operaciones booleanas entre miembros, de ello se encarga la función llamada "EVALUA2".

Esta función opera con la matriz *V_RED* y el vector *V_RES* arrojado por la función *EVALUA*. Esta función no entra en contacto con los datos de la base de datos, se basta con los datos contenidos hasta ese momento en el vector V_RES y ejecutando las operaciones indicadas en la matriz *V_RED*. Al final del proceso el valor final de la expresión se encontrará en la última casilla indicada en la matriz *V_RED*.

Continuando con el ejemplo, el vector *V_RES* al final de la función *EVALUA2* contendrá los siguientes valores :

V_RES

No. Renglón	Valor
1	1
2	0
3	0
4	0
5	1
6	0
7	0

esto es, si trabajamos con el 2 renglón de la matriz *V_RED*, la función *EVALUA2* realizará el siguiente proceso : toma el valor de la casilla número 3 del vector *V_RES* y ejecuta una operación booleana (*OR*) con el valor de la casilla número *4* del vector V_RES y el resultado lo almacena en la casilla número *6* del vector *V_RES*. De esta misma forma continúa

trabajando con los renglones restantes de la matriz *V_RED*, al final regresa el último valor obtenido en el vector *V_RES* y como se observa en este ejemplo el registro no cumplió con la expresión dada. Este proceso de compilación para expresiones query viene resumido en el siguiente diagrama.

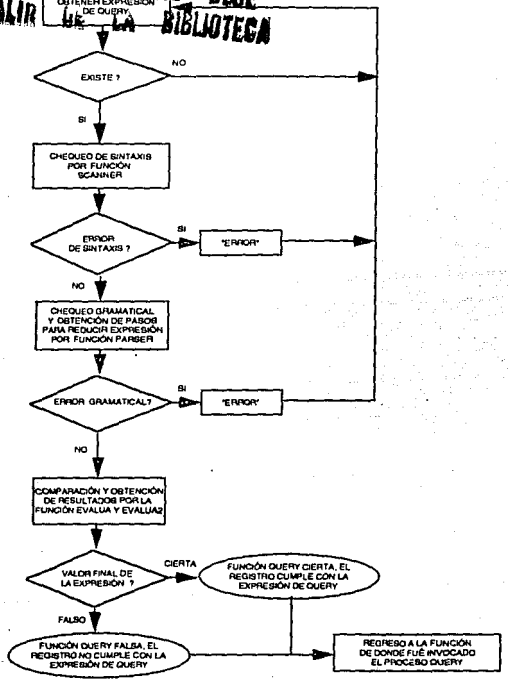


DIAGRAMA DE FLUJO PARA EL PROCESO DE COMPILACIÓN DE EXPRESIONES QUERY

CAPITULO VII

IMPORTACION EXPORTACION

Soluciones llenan los océanos,
soluciones corren por nuestras venas.

VII. IMPORTACION-EXPORTACION

Durante mucho tiempo se manejaron procesos y sistemas propietarios muy cerrados, de tal manera que cada proveedor desarrollaba su propio estándar en el mercado y ofrecía éste como una alternativa en el desarrollo de ciertas aplicaciones, lo cual provocaba el tener grandes sistemas aislados. Posteriormente surgió la idea de tener ambientes compatibles bajo el concepto de "Open Systems". La idea fué genial y la mayoría de fabricantes de Software y Hardware, comenzaron a tener acuerdos para lograr una mayor integración entre ambas partes, ofreciendo a sus clientes una alternativa (entre tantas) de integración para sus sistemas existentes a los sistemas actuales y una plataforma de desarrollo soportada por una gran cantidad de firmas.

El interés por tener compatibilidad de sus aplicaciones con el mundo fue tan grande que un usuario llegaba a adquirir un sistema (fuera de Hardware o Software) por las recomendaciones de compatibilidad que el proveedor remarcaba y esto solo se lograba mediante importaciones y exportaciones (conversiones) de formatos.

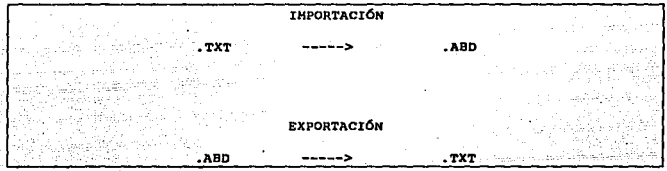
En 1990 con la aparición de Windows 3.0 se abrió una nueva opción, se ofrecía un ambiente multiproceso con la posibilidad de que las aplicaciones corriendo bajo este ambiente compartieran sus datos de una manera nueva de la ya conocida (conversión entre formatos). Microsoft decidió llamar a esta característica DDE (Dinamic Data Exchange) y DDL (Dinamic Data Link). Así pues, todas las aplicaciones podrían no solamente compartir datos sino además mantener una liga real entre los datos de diferentes aplicaciones, de tal modo que un dato que era modificado en una aplicación "A", era modificado también en una aplicación "B" automáticamente, si el dato era el mismo y si existía una liga entre las dos aplicaciones.

El caso de AVE no se aparta de estas dos formas de ligar datos con aplicaciones ya que cuenta con la facilidad de usar datos desde cualquier aplicación corriendo bajo Windows 3.0, es decir cuenta con la característica natural proporcionada por Windows 3.0 llamada DDE. Así también cuenta con la característica de compartir datos (importación y exportación) con cualquier otra aplicación corriendo bajo DOS o Windows. Esto se logra mediante una conversión de formatos, es decir, se convierte de formato AVE a formato texto separando los campos de un registro por comas y cada registro de otro por un CR (Carry Return) y un LF (Line Feed).

Se decidió utilizar este último formato por ser reconocido por el manejador de datos más conocido en el mercado : DBASE y al tener liga con este manejador, aseguramos que la tendrá con la mayoría de los sistemas existentes en el mercado.

Como característica adicional, el manejador de archivos AVE, tiene la facilidad de importar en un archivo de formato AVE existente o uno nuevo, de tal forma que se logra obtener un "append" o un "create" (según sea el caso) implícito en el sistema AVE. Así también existe otra característica adicional : al exportar un archivo de formato AVE al formato texto, existe la posibilidad de aplicar filtros (query), de tal forma que el usuario podrá exportar la información que en realidad le interese y no todo el archivo AVE.

El manejador de archivo AVE, asume dos tipos de extensiones para poder realizar la importación ó exportación, estas extensiones podrán ser cambiadas si es que el usuario lo desea, en caso contrario podrá trabajar con las extensiones de default : .TXT para archivos de texto y .ABD para archivos de bases de datos de formato AVE, tal como se muestra en el siguiente diagrama :


IMPORTACIÓN
.TXT
----->
.ABD
EXPORTACIÓN
.ABD
----->
.TXT

CAPITULO VIII

AMBIENTE DE PROGRAMACION WINDOWS

Una cosa es alcanzar la fortuna,
y otra merecer.

VIII. AMBIENTE DE PROGRAMACION WINDOWS 3.0

En los siguientes párrafos se explicarán las características y bondades principales que ofrece la programación en Windows 3.0, en comparación con la programación en lenguaje "C" estándar.

Las aplicaciones Windows son muy diferentes a las aplicaciones convencionales (en lenguaje estándar "C"). Esto es, porque para correr exitosamente en el ambiente Windows, se debe de cooperar con otras aplicaciones además de Windows, dejando que este último tenga el mayor control posible sobre los recursos del equipo y de las aplicaciones.

De lo anterior se deduce que las aplicaciones que se desarrollan bajo el ambiente de Windows son o deben ser mas complejas que las tradicionales y esto es entendible por las siguientes razones :

- Por ofrecer una interface gráfica bajo ventanas, cajas de diálogo y controles.
- Entradas y Salidas al sistema en base a colas.
- Gráficas en base a dispositivos independientes.
- Intercambio de datos entre aplicaciones (DDE).

La mayoría de los procesos bajo "C" estándar utilizan las librerías estándar para tener posibilidades de Entrada y Salida (E/S) al programa y manejo de memoria entre otras actividades. Estas librerías asumen un ambiente estándar de operación consistente de E/S en base a terminales con interface texto, además, como un acceso exclusivo a la memoria del sistema para E/S a los dispositivos de la computadora. En Windows, estas bases no son válidas, ya que las aplicaciones comparten los recursos de

la máquina incluyendo el CPU, y las formas de entrada son en base a un display gráfico, teclado y ratón.

VIII.1. INTERFACE DEL USUARIO DE WINDOWS 3.0

Una de las principales metas de Windows es proporcionar acceso visual a la mayoría de las aplicaciones al mismo tiempo en un ambiente multitareas, entonces es importante proporcionar una parte de la pantalla a todas esas aplicaciones, asegurando que el usuario pueda interactuar con todas esas aplicaciones que se encuentran corriendo. Esta interacción se hace posible por medio de una combinación visual de controles tales como menúes, controles y barras de scroll dentro de una ventana y que el usuario podrá utilizar para control de alguna aplicación.

En "C" estándar, el sistema prepara automáticamente la pantalla para una aplicación, típicamente, este pasa un handle de archivo a la aplicación, entonces se usa el handle para enviar cualquier E/S a la pantalla usando las librerías estándar de "C" para DOS. En Windows, uno debe de crear su propia ventana antes de poder tener cualquier E/S. Una vez que se ha generado la ventana para alguna aplicación, Windows porporciona una gran cantidad de información acerca de lo que el usuario está haciendo con dicha ventana, y ejecuta automáticamente cualquier proceso que el usuario requiera, tal como mover y el cambiar de tamaño a una ventana. Otra ventaja de desarrollar en Windows es que en contraste con los programas estándar de "C", el cual accesa una sola pantalla a la vez, una aplicación Windows, puede crear y usar cualquier número de ventanas para desplegar información en varios caminos. Entonces se podría decir que Windows maneja la pantalla por uno y se asegura de que dos aplicaciones no intenten accesar la misma parte de la pantalla al mismo tiempo.

VIII.2. COLA DE ENTRADA

Una de las grandes diferencias entre las aplicaciones Windows y de "C" estándar es la forma en que la aplicación recibe la entrada del usuario; en "C" estándar, una programa lee del teclado por medio de una llamada explícita a una función, tal como getchar en "C". La función típicamente espera hasta que el usuario presione una tecla y después regresa el código del carácter a el programa. En Windows, una aplicación no realiza llamadas explícitas para leer una entrada del teclado, en cambio, recibe todas las entradas del teclado, ratón, y del timer en la cola del sistema, y automáticamente redirecciona la entrada a la aplicación copiando esta desde la cola del sistema a la cola de la aplicación. Cuando la aplicación esta lista para recibir una entrada, esta lee un mensaje de entrada de su propia cola y despacha el mensaje a la ventana de la aplicación apropiada.

Windows proporciona automáticamente entrada desde el teclado y ratón para cada ventana. La entrada viene en un formato no uniforme llamada "input message". Un mensaje de entrada de una ventana contiene información que excede al tipo de información disponible desde "C" estándar, la cual únicamente proporciona 8 bits de información por cada carácter, en cambio, en Windows un mensaje de entrada contiene mayor información como es la hora del sistema, la posición del ratón, el estado del teclado, el código de cada tecla presionada, el botón del ratón presionado y el tipo del dispositivo que generó el mensaje. Por ejemplo, existen dos tipos de mensajes para indicar que una tecla fue presionada y liberada (WM_KEYDOWN y WM_KEYUP).

Por último Windows procesa de la misma manera los mensajes proporcionados por el teclado, ratón y timer.

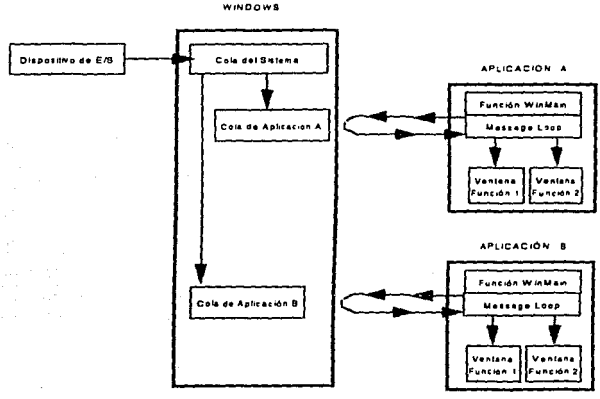
VIII.3. MULTIPROCESO

Windows es un sistema multiproceso, esto significa que más de una aplicación podrá correr al mismo tiempo. En el ambiente estándar de "C", no se proporciona multitarea ya que la mayoría de los programas asumen que ellos tienen control exclusivo de todos los recursos en la computadora, incluyendo los dispositivos de E/S, memoria, pantalla del sistema, y del CPU. En Windows, cualquier aplicación debe de compartir estos valiosos recursos con todas la aplicaciones que están actualmente corriendo. Por esta razón, Windows controla cuidadosamente estos recursos.

VIII.4. EL MESSAGE LOOP

Desde que una aplicación recibe una entrada a través de una cola de alguna aplicación, el "Message Loop" gobierna cualquier operación bajo el ambiente Windows. El Message Loop toma todos los mensajes de entrada de todas las aplicaciones y despacha estos a la ventana adecuada.

La siguiente figura muestra como Windows y cualquier aplicación colaboran entre sí para procesar un mensaje de entrada desde cualquier dispositivo de entrada (Hardware). En este caso en particular ejemplificaremos con un mensaje producido desde el teclado.



Windows recibe la entrada del teclado cuando el usuario presiona y libera alguna tecla. Windows entonces copia el mensaje del teclado desde la cola del sistema a la cola de la aplicación. El "Message Loop" recupera el mensaje del teclado de la cola de la aplicación, convierte este a un mensaje de código ANSI (WM_CHAR), y despacha este mensaje ANSI, así como el mensaje del teclado hacia la ventana o función adecuada. La ventana entonces usará este mensaje para los fines que se requieran.

El ciclo de vida de un "Message Loop" está definido por la creación de alguna ventana y un ciclo finito dentro del cual únicamente se captan y envían mensajes hacia dicha ventana, cuando esta ventana se cierra entonces se envía un mensaje de terminación hacia el "Message Loop" indicando que el ciclo (Message Loop) ha terminado.

VIII.5. CONSTRUCCION DE UNA APLICACION WINDOWS

Una aplicación Windows se puede diseñar usando cualquier editor ASCII (en este caso se utilizó Brief Editor) en combinación con la lista de software y hardware listado a continuación :

VIII.6. Software :

- Microsoft C Optimizing Compiler : CL v 6.0
- Microsoft Segmented-Executable Linker : LINK v5.0
- Microsoft Windows Resource Compiler : RC
- Microsoft Windows SDK Paintbrush : SDKPAINT
- Microsoft Windows Dialog Editor : DIALOG
- Brief Editor
- DOS 4.0 o 5.0
- Programa de Mantenimiento : MAKE

VIII.7. Hardware :

- Microcomputadora HP QS/16 386 con 3 Mbytes de Memoria Extendida como mínimo.
- Monitor VGA
- Disco Duro de 40 MB
- Unidad de disco flexible de 51/4 alta densidad
- Impresora HP Laser Jet III.

VIII.8. El procedimiento para la obtención del archivo ejecutable para el ambiente de Windows es el siguiente :

1. Usando un editor de texto se genera el archivo fuente en lenguaje "C" ó ensamblador (si el caso lo requiere) usando las funciones y librerías propias de Windows. Este programa es el que contendrá el programa principal WinMain, además de todas las funciones usadas.

2. Se generan los iconos, bitmaps y las cajas de diálogo a usar, esto se logra utilizando los editores de recursos (SDKPAINT, DIALOG y FONTEDIT).
3. Se genera un archivo de recursos (.RC) en el cual se definen todos los recursos a utilizar. En este archivo también se definen los menúes, cajas de diálogo, y otros recursos como tablas de cadenas.
4. Usando el RC (Resource Compiler) con la opción "-r" se compila el archivo ".RC" para obtener un archivo binario de recursos (.RES).
5. Se genera el módulo de definición de ambiente para Windows en un archivo texto con extensión (.DEF).
6. Se compilan todos los programas fuentes en lenguaje "C" con el compilador de Microsoft C versión 6.0 (CL) y/o todos los programas fuentes en ensamblador usando el ensamblador MASM de la compañía Microsoft.
7. Usando el ligador (LINK), se ligan todos los archivos objeto (.OBJ) previamente compilados con CL o MASM según sea el caso. Este procedimiento generará un archivo ejecutable (.EXE), pero este no se podrá ejecutar todavía, ya que no tiene incluidos los archivos de recursos generados.
8. Con el compilador de recursos (RC) sin la opción "-r" se agregan los archivos binarios de recursos (.RES) al archivo ejecutable generado (.EXE).
9. El archivo generado en el paso anterior contiene también la extensión .EXE y es este el que podrá ser ejecutado desde Windows.
10. Se afinan y corrigen los errores encontrados.

En la siguiente figura se ilustran gráficamente los pasos a seguir:

Crear archivos fuentes

Generar archivos de Recursos

Generar el Archivo Script de Recursos

Compilar los archivos fuente

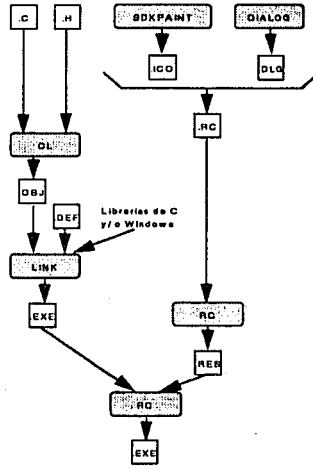
Crear el Archivo de Definición .DEF

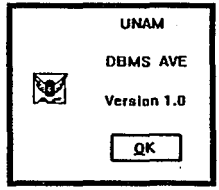
Ligar los módulos objeto con librerías de C y Windows.

Compilar los archivos de recursos

Agregar los recursos al archivo executable

El resultado es una aplicación Windows 3.0





MANUAL DE REFERENCIA DEL USUARIO

DBMS AVE

Sistema Manejador de Base de Datos

Facultad de Ingeniería
Universidad Nacional Autónoma de México.

INDICE

CAPITULO IX

MANUAL DEL USUARIO

Comprender la felicidad de que el suelo sobre el que estas de pie no puede ser mas grande que los dos pies que lo cubren.

IX.1. INTRODUCCION.

Qué es el DBMS AVE ?

Es un Sistema Manejador de Archivos para microcomputadoras PC y compatibles, que se encuentra desarrollado para el ambiente de programación Windows y por tanto posee todas las ventajas y facilidades propias de ante ambiente.

El *DBMS AVE* permite realizar las operaciones de creación, organización, almacenamiento y recuperación de información sobre bases de datos. Para ello podemos decir que una parte del sistema está dirigida al manejo de las estructuras de las bases de datos, la cual incluye la creación , modificación, consulta y eliminación de estructuras. Por otro lado, tenemos el manejo de las operaciones correspondientes a la información contenida en nuestra base de datos, las cuales pueden ser clasificadas como operaciones a nivel registro y operaciones a nivel grupo de registros. Dentro del primer tipo podemos encontrar las operaciones de agregar, editar, borrar registros y dentro del segundo tipo podemos citar las operaciones de ordenar, indexar, compactar, querys, importación y exportación de registros. Además, el sistema cuenta con algunas utilerías que permiten entre otras cosas la definición de ambiente del sistema, la facilidad de tener acceso a comandos del Sistema Operativo, etc.

Como anteriormente se mencionó el hecho de ser una aplicación Windows nos permite tener a nuestro alcance las facilidades propias del ambiente como lo son: el manejo de la multitarea , es decir tener acceso a varias aplicaciones al mismo tiempo, aquí cabe señalar que es posible ejecutar el *DBMS AVE* repetidamente y donde el número de veces que lo podemos ejecutar depende del total de memoria disponible. Por otro lado también el manejo dinámico de la memoria, y las facilidades en la interface con ventanas.

Contenido del Sistema.

El Sistema *DBMS AVE* está compuesto por los siguientes archivos:

Ejecutables:

AVE.EXE

Datos:

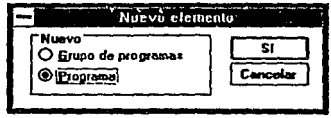
ADA.HDR	Archivo Descriptor de Archivos.
*.ABD	Archivos de Base de Datos que se van generando.
AVE.DAT	Catálogo de Errrores.

IX.2. INSTALACION.

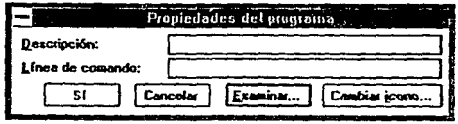
Para instalar la aplicación AVE se procede de la siguiente forma:

1. Ejecutamos Windows, nos posicionamos en el grupo de trabajo en el cual deseamos instalar la aplicación, se recomienda que se instale en el grupo correspondiente a las aplicaciones Windows.

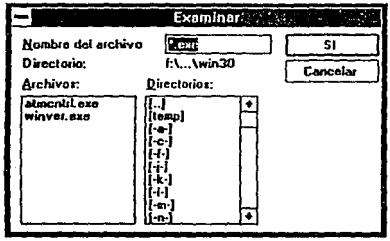
2. Seleccionamos del menú principal la opción "ARCHIVOS" y de aquí la opción de "Nuevo", con lo que aparece el siguiente despliegue en pantalla:.



3. De aquí seleccionamos la opción "Programa" con lo que obtenemos:



4. De aquí seleccionamos la opción de "Examinar" donde elegimos la trayectoria y nombre de la aplicación a instalar, en este caso AVE.EXE.



5. Una vez seleccionado se procede a su instalación la cual se establece a través de la aparición del icono siguiente:

IX.3. ACTIVACION DEL SISTEMA AVE.

Para poder iniciar la ejecución del Sistema AVE realizamos lo siguiente:

1. Ejecutamos Windows y nos posicionamos en el grupo de trabajo en el cual se encuentra la aplicación AVE.

2. Seleccionamos el icono de AVE y pulsamos dos veces el botón izquierdo del mouse o bien la tecla de ENTER. El icono de AVE es el siguiente:

IX.4. MENU PRINCIPAL.

Una vez que el símbolo (icono) asociado con el sistema AVE ha sido seleccionado, se hace evidente una ventana similar a las que presenta Windows.

La figura 4.1 es una réplica del menú principal del sistema AVE en el cual se distingue lo siguiente :

a)En la parte superior y al centro se observa el distintivo DBMS AVE, el cual hace referencia al nombre del manejador de bases de datos conocido como AVE.

b)A continuación, en el siguiente renglón de la ventana se distinguen las cinco opciones de carácter general consideradas por el sistema.

Tales opciones se enumeran a continuación:

-Archivos
-Registros
-Operaciones relacionales
-Utilería
-Sistema

Para activar una de tales opciones es suficiente colocar el cursor sobre el nombre de la elegida y dar un "click" izquierdo con el

IX.4. MENU PRINCIPAL.

Una vez que el símbolo (icono) asociado con el sistema AVE ha sido seleccionado, se hace evidente una ventana similar a las que presenta Windows.

La figura 4.1 es una réplica del menú principal del sistema AVE en el cual se distingue lo siguiente :

a)En la parte superior y al centro se observa el distintivo DBMS AVE, el cual hace referencia al nombre del manejador de bases de datos conocido como AVE.

b)A continuación, en el siguiente renglón de la ventana se distinguen las cinco opciones de carácter general consideradas por el sistema.

Tales opciones se enumeran a continuación:

-Archivos
-Registros
-Operaciones relacionales
-Utilería
-Sistema

Para activar una de tales opciones es suficiente colocar el cursor sobre el nombre de la elegida y dar un "click" izquierdo con el

ratón.Cada una de las opciones tiene un menú de posibilidades las cuales serán exploradas en los capítulos siguientes.

Por otra parte, la ventana del menú principal, como cualquier otra aplicación de Windows, puede ser colocada en otra región de la pantalla y así también ser modificado su tamaño, si se desea.

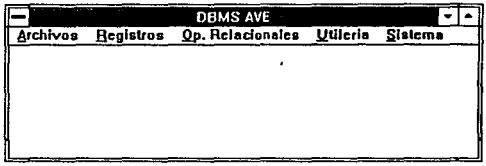
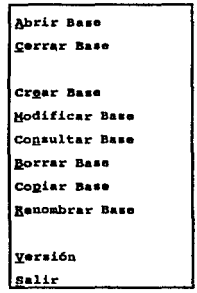


FIGURA 4.1 MENU PRINCIPAL.

IX.5. SUBMENUES.DEL SISTEMA

En el capítulo anterior se mencionó el modo de activar una determinada opción del menú principal, su efecto se observa en cada una de las cinco ventanas de bienvenida que a continuación se presentan :

Archivos:

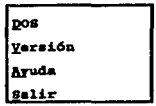


Registros:

Operaciones Relacionales:

Utilería:

Sistema:



En ellas se distinguen los siguientes rasgos comunes :

a)Que cada opción del menú principal tiene a su vez un conjunto de posibilidades (un submenú) observables dentro de un recuadro propio a cada opción.

b)Que cuando una opción del menú principal es seleccionada aparece en una tonalidad que la resalta de las demás; de igual manera se observa la primera posibilidad en el submenú respectivo.

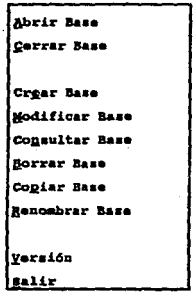
c)Que cada submenú desaparece al dar un "click" cuando se tiene el cursor colocado sobre el nombre de la opción, o bien, oprimiendo el botón izquierdo del ratón fuera del recuadro asociado al submenú.

d)Que si no hay algún submenú activo, entonces, el usuario esta situado frente al menú principal del sistema.

IX.6. DESCRIPCION DE SUBMENUES.

IX.6.1 SUBMENU DE OPCION ARCHIVOS.

Cada vez que se elige la opción "ARCHIVOS" del menú principal, se despliega su ventana y sus funciones semejante al gráfico anexado:



Por lo que resta, se hará una descripción una a una de las funciones incluídas indicando su propósito y la manera como interactúa con el usuario, esto es, su uso.

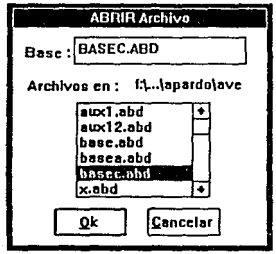
Como primer punto la activación de cualquier función en cualquier submenú se lleva a cabo colocando el cursor sobre el nombre de la función de interés y a continuación se presiona el botón izquierdo del ratón una sola vez. Dependiendo de la función escogida, el sistema responde con un mensaje en algunas ocasiones, la mayor de las veces lo hace exhibiendo una ventana bastante elaborada donde al usuario se le inquiere por información adicional.

Comencemos...

FUNCION ABRIR BASE.

Propósito: Antes de arrancar una sesión que involucre el acceso a un archivo es conveniente que dicho archivo se halle abierto.

Al activar esta función se hace presente la siguiente ventana :



Obsérvense los dos botones inferiores: uno de ellos corresponde a la acción de aceptar (OK), y el otro al de cancelar el uso de esta función, además de que nos devuelve al menú principal.

Hay dos maneras de especificar el archivo de datos que se desea poner en marcha (abrirlo) y cada una de ellas relacionada con uno de los recuadros que se observan.

1)PRIMER METODO

- Llévese el cursor al interior del recuadro identificado por la palabra base y presione el botón izquierdo del ratón una vez.

- Proporciónese el nombre del archivo, indicando su extensión(.ABD) y trayectoria.

- Seleccione el botón de aceptación (OK).

2)SEGUNDO METODO

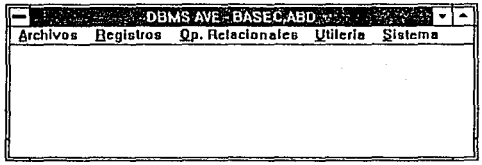
- Observe el recuadro inferior y note la presencia de nombres de archivos con extensión .ABD (si los hay) y de letras que especifican unidades de disco.

- Los archivos mencionados (si los hay) se encuentran bajo el subdirectorio que se indica en la línea que dice "ARCHIVOS EN:"

- Si el archivo que desea abrir se encuentra en esta lista, mueva el cursor sobre el nombre y oprima el botón izquierdo del ratón una vez, como resultado, una copia del nombre del archivo se traslada al recuadro superior.
- Si está de acuerdo con su elección, active el botón de aceptación y eso es todo.

- Si el caso es que el archivo se localiza en otra unidad de disco, seleccione la letra correspondiente a la unidad y note que aparece dentro del recuadro superior, a continuación proceda de acuerdo a lo expuesto en el primer método.

Advierta que una vez que se ha activado un archivo de base de datos, el submenú desaparece y el sistema nos guía de nueva cuenta al menú principal, note que el encabezado de la ventana principal se ha modificado: no tan solo se lee el nombre del manejador sino también el nombre del archivo que está en condición de ser manejado, como lo muestra la siguiente pantalla.



FUNCION CERRAR BASE.

Propósito: Cerrar el archivo de base de datos que esté abierto.

Procedimiento:

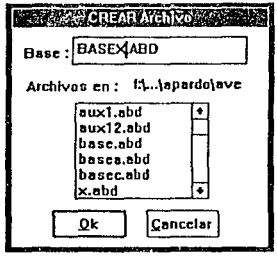
- Coloque el cursor sobre el nombre de la función y oprima el botón izquierdo del ratón una sola vez.
- El archivo quedará cerrado y no es posible accesarlo más.
- El sistema retorna al menú principal.

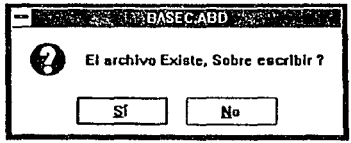
FUNCION CREAR BASE.

Finalidad: Crear un descriptor para un archivo de base de datos y hacerlo parte integral del conjunto de archivos que administra el sistema.

Active la función y observe la pantalla que aparece en respuesta, claramente se nota su semejanza con la mostrada en la función "ABRIR BASE", excepto por el encabezado superior que identifica la operación a realizar. En efecto, opera de manera similar

salvo cuando se desea dar de alta un descriptor que ya existe, si éste es el caso, el sistema preguntará si se desea sobreescribir. Se ejemplifica a continuación lo expuesto :





Si la decisión es la de no sobreescribir, la ventana de presentación permanece vigente, dando la oportunidad de indicar algún otro archivo que no esté dado de alta.

En las líneas siguientes se muestra el camino para la creación del descriptor de un archivo de base de datos.

a) Seleccione la unidad de disco donde se alojará el archivo, para tal fin, coloque el cursor en alguno de los especificadores de disco localizados en el recuadro inferior y a continuación oprima el botón izquierdo del ratón una vez.

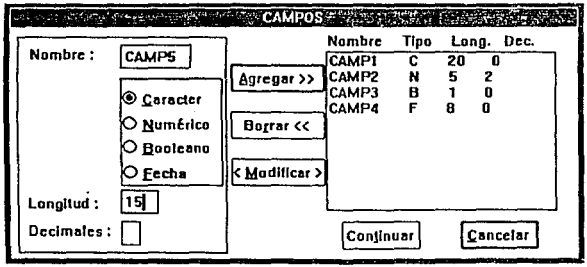
b) Observe el recuadro superior y compruebe que aparece la letra seleccionada.

c) Mueva el cursor dentro del recuadro superior y oprima el botón izquierdo del ratón.

d) Borre los caracteres no deseados y proporcione el nombre del archivo usando el teclado.

e) Mueva el cursor sobre el botón de aceptación (OK) y oprima el botón izquierdo del mouse.

f) El archivo queda creado e inmediatamente emerge una nueva ventana :



Ahora, el sistema está en posición de aceptar las características de los campos de datos que mantendrá el descriptor del archivo; así también es posible que las modifiquemos o borremos si así lo deseásemos.

Consideremos las operaciones de agregar, borrar y modificar los campos y sus atributos que definen la estructura de un archivo del sistema.

SECUENCIA PARA AGREGAR UN NUEVO CAMPO

a) Mueva el cursor al interior del recuadro superior izquierdo, oprima el botón izquierdo del ratón una vez y escriba el nombre de un campo de hasta cinco caracteres.

b) Lleve el cursor al recuadro inferior y seleccione el tipo del campo oprimiendo una vez el botón izquierdo del ratón.

c) Especifique la longitud del campo así: coloque el cursor en el área donde se solicita tal parámetro y presione el botón izquierdo del ratón, entonces indique la longitud como un número entero. Algunos campos no requieren de este dato.

d) Para campos numéricos, indique también el número de cifras decimales a mantener en el dato.

e) Translade el cursor sobre el botón de agregar y presione el botón izquierdo del ratón, como resultado, el nombre del campo y sus atributos se copian al recuadro derecho.

f) Observe que cada vez que se adiciona un nuevo campo en el recuadro derecho aparecerá ordenado alfabéticamente.

SECUENCIA PARA MODIFICAR UN CAMPO

Si se requiere corregir alguna característica de un campo ya especificado se sugieren los siguientes pasos :

a) Deslice el cursor hacia el recuadro donde aparecen los campos y sus características.

b) Posicione el cursor sobre el renglón donde está el campo a modificar.

c) Marque el renglón presionando el botón izquierdo del ratón y observe que cambia de tonalidad.

d) Seleccione el botón de "MODIFICAR", verifique que el campo en cuestión junto con sus atributos abandona su posición actual y se translada al recuadro de la izquierda.

e) Ahora puede hacer los cambios que deseé.

f) Una vez que los ajustes se hayan hecho proceda de acuerdo a la secuencia de "AGREGAR".

SECUENCIA PARA BORRAR UN CAMPO

a) Estando el cursor en el recuadro derecho marque el campo a cancelar.

b) Elija el botón de "BORRAR" y el campo desaparece, es todo.

Cuando todos los campos hayan sido definidos y se deseen salvar como parte de la estructura de un archivo lo único que resta es seleccionar el botón de "CONTINUAR", hecho lo anterior estaremos de regreso al menú principal.

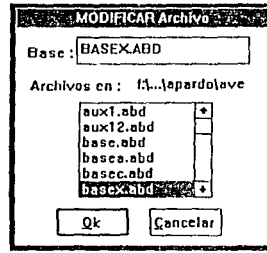
Para anular todo lo estipulado, si así lo requiere, marque el botón de "CANCELAR".

FUNCION MODIFICAR BASE.

La estructura que define un archivo de base de datos (dada de alta) puede ser alterada a través de esta opción. Para lograr tal efecto haga lo siguiente :

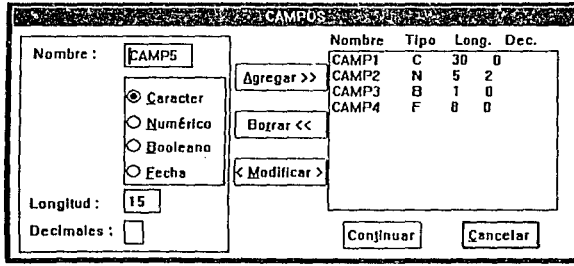
a) Active la función.

b) El sistema responderá con una ventana que ya es conocida, seleccione el archivo a modificar vía ratón o bien especifíquelo en el recuadro superior de manera manual vía teclado.



c) Marque el botón de aceptación (OK).

d) Observe la ventana que brota, si recuerda, es idéntica a la que acompaña a la función "CREAR BASE", con la única aclaración de que ahora ninguno de los tipos de los datos aparece marcado.



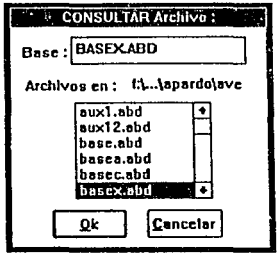
e) Para agregar, borrar o bien modificar, proceda de manera similar como en la función "CREAR BASE" según sus secuencias ahí tratadas.

FUNCION CONSULTAR BASE.

Objetivo: Visualizar la estructura que tiene la información de un archivo de base de datos.

Al igual que con las funciones anteriores, actívela !.

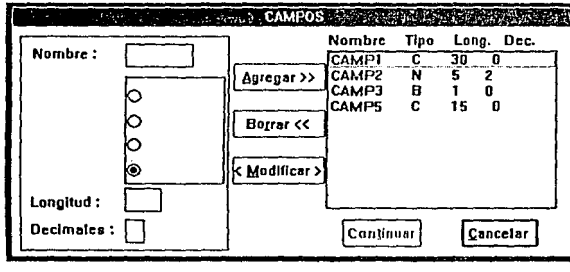
La ventana que aparece sirve para solicitarle al sistema que nos muestre la estructura del archivo cuyo nombre hemos indicado.



Procedemos de la siguiente manera:

Seleccione un archivo.

La ventana que surge muestra los nombres de los campos y sus características asociadas a cada uno de ellos, y por recordarlo, creo que ya se está acostumbrado a ese tipo de representación, sin embargo puede que sea engañosa :



En primer lugar, observe que los tipos de los datos en el recuadro izquierdo se muestran en un tono pálido, síntoma de que no hay posibilidad de activarlos por medio del ratón, lo mismo le pasa a la tecla de "CONTINUAR", si lo duda haga una prueba.

También, la acción de indicar el nombre de un nuevo campo, su longitud y el número de decimales, no está permitida. Puede oprimir el botón de "AGREGAR" ya que su acceso no está negado, pero no se realizará.

Como se observa, los botones de BORRAR y MODIFICAR tampoco están deshabilitados, aún más, es posible el uso del cursor y al hacerlo poder marcar un campo y borrarlo, hagalo !, de igual modo es posible realizar el intento de modificarlo y después agregarlo, lo anterior aparentemente ocurre sin embargo tales alteraciones no son efectivas ya que el botón de CONTINUAR esta inactivo.De manera que la única posibilidad efectiva que nos queda es la de CANCELAR la cual nos regresará al menú principal del sistema.

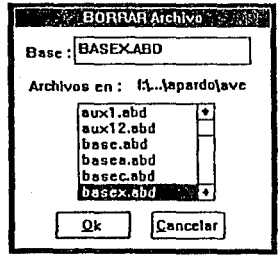
En conclusión, el descriptor del archivo consultado permanece intacto.

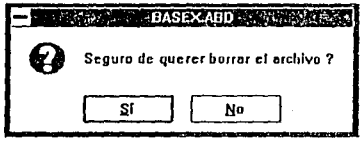
FUNCION BORRAR BASE.

Finalidad: Descargar del sistema un archivo de base de datos al igual que su descriptor.

Se logra de la siguiente forma:

En análoga manera que en los casos anteriores, el sistema responde con una ventana de saludo en la cual tenemos la oportunidad de seleccionar o indicar el nombre del archivo que deseamos cancelar. Haga el intento indicando un archivo. Observe que al instante de que selecciona el botón de aceptación (OK) brota una ventana que muestra en su encabezado el nombre del archivo y se nos pide confirmar nuestra decisión.





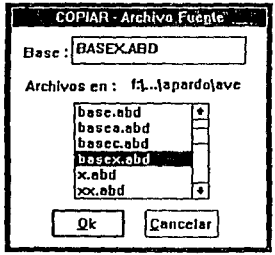
FUNCION COPIAR BASE.

Propósito: Obtener un duplicado de la estructura que define a los datos de un archivo de base de datos.

Procedimiento:

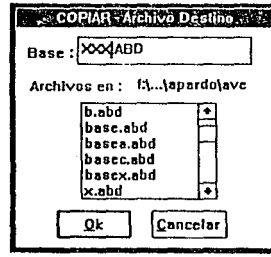
a) Active la función "COPIAR BASE".

b) Observe que emerge una ventana como esta:



c) La ventana anterior tiene como fin el indicar o seleccionar un archivo cuya estructura se desea duplicar. Elija uno y use el ratón para activar el botón de selección.

d) En la ventana que a continuación surge se tiene que especificar el nombre del archivo que tendrá una estructura similar al archivo indicado en el inciso anterior. Elija un nombre y marque el botón de aceptación para finalizar el proceso.



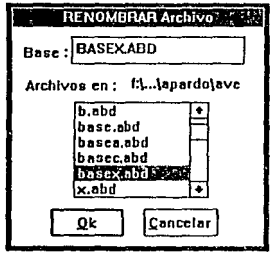
FUNCION RENOMBRAR BASE.

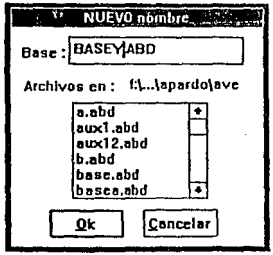
Objetivo: Cambiar el nombre a un archivo de base de datos.

Veamos como opera :

a) Active la función.

b) De la ventana que se presenta indique o elija el nombre del archivo. Hágalo y oprima el botón de selección.



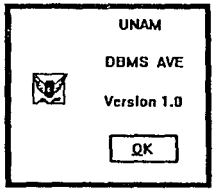


c) Si desea checar la modificación vaya a DOS y consulte el directorio.

FUNCION VERSION.

Propósito: Desplegar la versión y ventana alusiva al sistema.

Actívela.

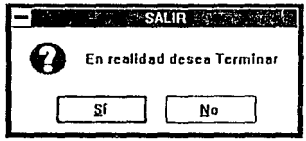


Para regresar al menú principal proceda al igual que con las funciones ya tratadas.

FUNCION SALIR.

Finalidad: Abandonar el sistema AVE.

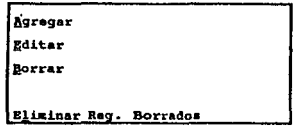
Actívela. La ventana que se hace evidente requiere de nuestra confirmación.



IX.6.1 SUBMENU DE OPCION REGISTROS.

Una vez definida la estructura de la información de un cierto archivo de base de datos, el paso natural a seguir es el de iniciar la captura de los datos acorde a los campos especificados. En este sentido, la opción "REGISTROS" del manú principal no solamente ofrece la acción antes mencionada, como se detallará a continuación.

Active esta opción y verifique la presencia de una ventana similar a la siguiente:



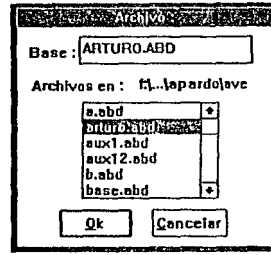
Las acciones que ahí se visualizan están relacionadas exclusivamente con el manejo de los registros de los archivos ligados al sistema y nos proponemos en los párrafos que restan, pormenorizar la operación de tales acciones.

FUNCION AGREGAR

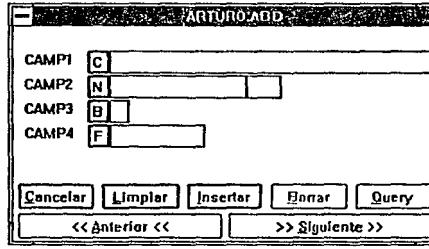
Objetivo: Insertar nuevos registros al final de un archivo de base de datos.

Active la función.

Si no hay algún archivo que se encuentre activo (abierto) surge a la vista una ventana como la siguiente:



Habrá a continuación que especificar el nombre del archivo al cual se le desea adicionar nueva información. Elija un archivo cuya estructura haya previamente definido y elija el botón de aceptación (OK). En respuesta, el sistema introduce una ventana :



La imagen anterior es nueva de manera que amerita tiempo para su observación y líneas de explicación para su entendimiento.

De manera general, sin considerar algún archivo de datos en especial, se percibe lo siguiente :

a) El nombre del archivo, con el cual se trabaja aparece como título de la ventana.

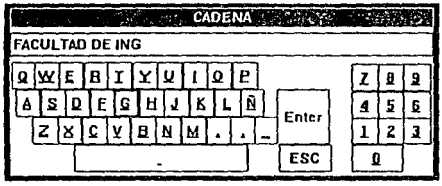
b) En la parte central se detectan los nombres de los campos del archivo, ordenados y acompañados cada uno de un recuadro a manera de renglón. Cada recuadro de tamaño acorde a su especificación. También, es notoria la presencia en cada renglón de un botón con una letra, que como es de suponerse realiza algún encargo. Cada letra de un botón guarda relación con el tipo del campo asociado :

B -- Dato tipo booleano.
C -- Dato tipo carácter.
F -- Dato tipo fecha.
N -- Dato tipo numérico.

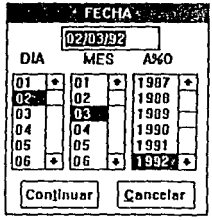
c) Active uno a uno los botones, brotarán las siguientes imágenes :

BOOLEANO:

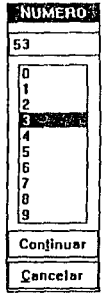
CARACTER:



FECHA:



NUMERO:



El propósito de estas ayudas es la de ofrecer una alternativa al teclado. Esto es, se pueden ir llenando los recuadros de cada campo seleccionando uno a uno : letras, números, caracteres especiales, valores booleanos o bien fechas, con tan solo ir marcando el carácter y cuando se haya completado la operación de formar el dato seguir con la activación del botón de "CONTINUAR" que aparece en cada una de esas ayudas.

Para el caso de los campos con tipos numérico, carácter o fecha, la selección de un carácter requiere que el botón izquierdo del ratón se presione dos veces. Note que cada vez que se elige un carácter, este aparece en la parte superior de la ventana de ayuda y que cuando se 'presiona' el botón de "CONTINUAR" la información formada se translada al recuadro del campo en proceso.

Si por alguna razón no desea que el dato así formado sea considerado como información válida, 'presione ' el botón de "CANCELAR"; la ventana y los datos se esfumarán.

Llene un campo booleano seleccionando su valor de verdad.

Por otra parte, lo tradicional es usar el teclado y el camino a seguir es bien conocido: En primer lugar, deslice el cursor al interior de cada renglón-recuadro, a continuación presione el ratón una vez, finalmente teclee los datos.

Al término de la captura de los datos de un registro el sistema ofrece tres caminos: cancelar, limpiar el registro editado o bien aceptar el dato y continuar con el proceso de agregación de información. Veamos como se desenvuelve lo antes mencionado :

En la parte inferior de la ventana se localizan siete botones con acciones indicadas, cuatro de ellos se presentan con una tonalidad más tenue lo que quiere decir que no están a disposición del usuario; los otros si lo están y tienen la siguiente interpretación :

Botón "CANCELAR": Usese para finalizar la tarea de agregar registros.

Botón "LIMPIAR" : Si parte de la información capturada no le satisface, con el uso de esta alternativa los datos del registro que se visualiza son desechados y las áreas de captación de datos se tornarán vacías.

Botón "INSERTAR": A fin de que el registro editado pase a formar parte del archivo de datos use esta opción, entonces, el sistema contestará con otro registro con sus campos limpios de información; en este punto, estamos en posición de generar un nuevo elemento del archivo.

FUNCION EDITAR

Finalidad: Desplegar los contenidos de los registros de un archivo de base de datos, registro por registro.

Active la función y seleccione alguno de los archivos de base de datos siguiendo el procedimiento ya conocido.

Observe detenidamente la ventana con que el sistema responde; idéntica a la de la función AGREGAR solo que se dispone de más botones habilitados.

Note también que como encabezado de la ventana aparecen el nombre del archivo que se está consultando y el número del registro visualizado.

Además, la apariencia del botón de INSERTAR señala que no es posible aumentar la cantidad de registros del archivo.

En las líneas a continuación, se detalla el uso de aquellos botones no considerados con anterioridad.

a) Botón "LIMPIAR": Tiene el mismo efecto visual como en AGREGAR pero ninguno sobre el archivo de datos. Como comprobación haga uso de esta opción, luego regrese al menú principal y active la opción de edición y cheque que la información permanece íntegra.

b) Botón <<ANTERIOR<<: Trae el contenido del registro previo al que se muestra. En el caso de solicitar la información del registro previo al primero, el sistema responde con un mensaje indicando tal eventualidad.

c) Botón >>SIGUIENTE>>: Presenta el contenido del registro siguiente al que tenemos en nuestra visión. Cuando se demanda el

registro siguiente al último, el sistema nos hace saber que hemos alcanzado el fin del archivo.

Los dos botones anteriores nos permiten desplazarnos dentro del archivo de base de datos y examinar sus contenidos registro a registro.

d) Botón "BORRAR": Facilita marcar un registro para su posible eliminación del archivo.

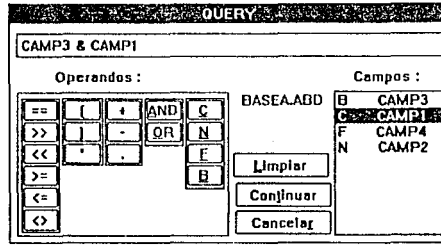
Elija un registro y seleccione el botón de borrado, observe que cambia la tonalidad que envuelve a los datos en el registro al igual que la coloración de los botones; dese cuenta que ahora el botón de BORRAR cambia de presentación: ahora se lee como botón de "RECUPERAR".

Advierta la benevolencia del sistema: si se arrepintió puede desmarcar el registro y recuperarlo; seleccione el botón de "RECUPERAR" y observe el regreso de la coloración original de los datos, es decir, el registro es aún parte del archivo de datos.

e) Botón QUERY: Para realizar alguna consulta selectiva a un archivo de base de datos use esta opción.

Pruebe !

Brotará la imagen siguiente:



Mire cuidadosamente esta presentación.

En la parte superior se distingue un renglón delimitado por un recuadro, en el podrá especificar alguna expresión que sugiera consulta alguna y que involucre y relacione campos del archivo con constantes alfabéticas, numéricas, de tipo fecha o lógicas, para ello, empleando operadores relacionales y lógicos.

Para construir una expresión como la mencionada, puede usar el teclado, sin embargo el sistema brinda ciertas facilidades, a saber: la presentación de operadores y comandos en un formato similar a la de una calculadora de bolsillo tal como se ve en un extremo de la ventana.

Note en el otro extremo los nombres de los campos del archivo.

A continuación se explica el camino para construir una expresión de consulta:

Para elegir un nombre de un campo, deslice el cursor sobre el nombre y oprima dos veces el ratón, el campo seleccionado aparecerá resaltado y será transferido al recuadro. Un operador se elige igual, solo que basta oprimir el ratón una vez.

Para crear constantes alfabéticas que deban estar entre comillas, el botón de comilla sencilla puede ser usado repetidamente.

Note que también están presentes los símbolos para operaciones lógicas de AND y OR; de igual modo se mantienen las ayudas para cada uno de los tipos de los campos y su manejo es idéntico al que se explicó en la función AGREGAR.

Por otro lado, si se equivocó o ya no es su deseo proseguir con la consulta, el botón de limpiar le será de utilidad ya que anula la petición. Por el contrario, si desea que la consulta se verifique, seleccione el botón de "CONTINUAR" y el sistema desplegará información acorde a la petición de consulta.

Para regresar a la ventana previa a la de consulta, active el botón de "CANCELAR".

FUNCION BORRAR

Propósito: Marcar aquellos registros no deseados para posteriormente ser dados de baja del archivo de base de datos.

Como siempre, active la función y revise la ventana de presentación, similar a las anteriores. Observe que los botones de "LIMPIAR" e "INSERTAR" son los únicos que permanecen inhibidos, los demás cumplen con sus funciones asignadas y que inclusive es posible realizar consultas.

Mire la información, los datos presentes se muestran en un tono tenue al igual que los botones adyacentes, inclusive, perciba que el cursor no tiene efecto sobre los renglones que contienen datos.

Ahora bien, para marcar un registro proceda así :

a) Con los botones "ANTERIOR" y "SIGUIENTE" busque el registro que desea marcar.

b) A continuación seleccione el botón de BORRAR y perciba que su nombre cambia al de "RECUPERAR", lo anterior tiene sentido en razón de que si cambia de idea, podrá desmarcar al registro.

c) Aún con los registros marcados es posible llevar a cabo de manera satisfactoria consultas pues, en realidad, no han sido eliminados.

d) Finalmente, para regresar al menú principal, "presione" el botón de CANCELAR.

FUNCION ELIMINAR REGISTROS

Objetivo: Dar de baja definitiva de un archivo de base de datos los registros marcados vía la función BORRAR.

Si no hay archivo de datos activo, seleccione uno.

Si el archivo elegido no tiene registros marcados, el sistema responderá con el siguiente mensaje:

En caso contrario, el sistema generará un mensaje de advertencia solicitando la confirmación de proseguir o cancelar la acción de la función.

IX.6.3 SUBMENU DE OPERACIONES RELACIONALES.

Este módulo contiene las operaciones que involucran un grupo de registros como lo son:

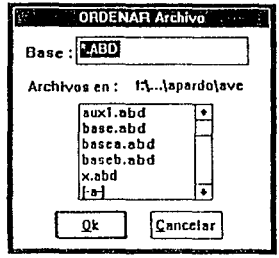
- ***Ordenamiento***
- ***Indexación***
- ***Query***

Descripción:

FUNCION ORDENAMIENTO.

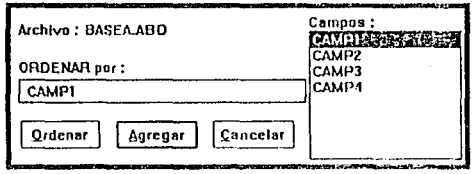
Esta opción permite realizar la creación de un archivo ordenado en forma ascendente y cuya precedencia está determinada por la expresión llave, misma que se define a partir de la selección de campos pertenecientes a la base de datos, de tal forma que el primer campo corresponde a la llave primaria y los demás campos son las llaves secundarias.

Para proceder con el ordenamiento primeramente hay que seleccionar la opción correspondiente dentro del menú con lo que se procede a verificar la existencia de un archivo de base de datos activo, el cuál en caso de no existir se solicita a través de la siguiente pantalla:



En esta pantalla como se observa podemos definir la trayectoria y nombre del archivo a activar ya sea manualmente en el primer renglón o bien podemos realizarla utilizando la ventana de selección colocada en la parte inferior con ayuda del ratón.

Una vez definido el archivo se procede a pulsar el botón de "OK" o bien pulsando dos veces el botón izquierdo del mouse. Acción que lleva la siguiente pantalla:

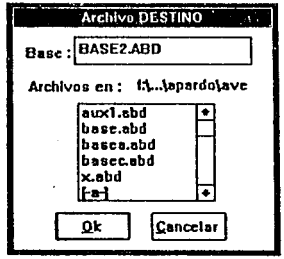


Donde el área rectangular ubicada a la izquierda bajo el letrero de "ORDENAR POR" corresponde al área de captura de la expresión sobre

la cual se va a realizar el ordenamiento. Dicha expresión esta formada por los nombres de los campos y puede ser definida manualmente y/o a través de la selección de la lista de campos a la derecha; para esta última acción únicamente nos posicionamos en el campo deseado y presionamos el botón de "AGREGAR".

Una vez que definimos la expresión llave procedemos a pulsar el botón de "ORDENAR" para realizar el ordenamiento o bien el botón de "CANCELAR" para abandonar el proceso.

En caso de seleccionar "ORDENAR" a continuación se solicita el nombre del archivo destino mediante la siguiente ventana:

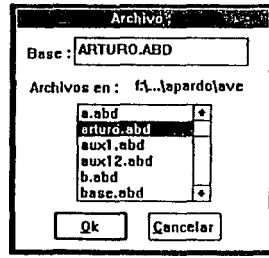


Y ya seleccionado el nombre se procede a ordenar.

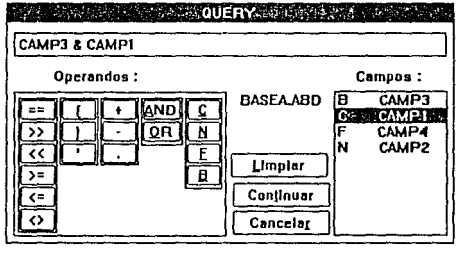
FUNCION QUERY.

Mediante esta opción es posible realizar el establecimiento de filtros ó vistas de información a través de la definición de condiciones en las que se involucran los campos de la base de datos, operadores relacionales, lógicos y de agrupamiento y valores definidos por el propio usuario. Una vez que se establece el filtro únicamente se tiene acceso a la información que cumple las condiciones establecidas.

Para realizar un query primeramente seleccionamos la opción "query" del menú de "Operaciones Relacionales" y al igual que las operaciones anteriores se realiza la verificación de un archivo de base de datos activo y de no existir se solicita a través de la siguiente pantalla:



Ya una vez activado el archivo con el que se desea trabajar aparece en pantalla la siguiente ventana:



Como se observa el primer renglón permite la captura de la expresión que establece las condiciones bajo las cuales se realizará el filtro de información.

Esta captura puede ser realizada manualmente y/o con la ayuda de las áreas de selección situadas en la parte inferior, en las cuales se encuentran por un lado los operadores lógicos, relacionales y de agrupamiento y por otro lado se tienen los campos pertenecientes a la base de datos actual.

Al finalizar la captura se procede a pulsar el botón de "OK" o bien pulsar dos veces el botón izquierdo del ratón, con lo que se inicia el proceso de validación de la expresión y en caso de haber error se notificará mediante el despliegue de un mensaje de error.

Una vez aceptada la expresión automáticamente queda establecido el filtro o vista en el manejo de la información, de tal forma que únicamente se tendrán activos los registros que cumplan con las condiciones establecidas.

Como observamos en la primer área titulada "Operandos" tenemos cinco columnas de botones mismas que describimos a continuación:

La primera columna corresponde a los botones con los operadores relacionales, entre los que encontramos:

==	IGUALDAD
>>	MAYOR QUE
<<	MENOR QUE
>=	MAYOR O IGUAL QUE
<=	MENOR O IGUAL QUE
<>, ><	DIFERENTE

La segunda columna pertenece a los botones de los operadores de agrupamento y al delimitador de cadenas de caracteres:

(	PARENTESIS DERECHO
)	PARENTESIS IZQUIERDO
'	DELIMITADOR DE CADENAS (EJ. 'BEETHOVEN')

La tercera columna contiene los botones correspondiente a los operadores aritméticos:

+	SUMA
-	RESTA
*	MULTIPLICACION

La cuarta columna de botones define a los operadores lógicos:

AND: Operación lógica que se rige con base a la siguiente tabla:

OPERANDO 1	OPERANDO 2	RESULTADO
VERDADERO	V	V
V	F	F
FALSO	V	F
F	F	F

OR : Operación lógica que tiene las siguientes características:

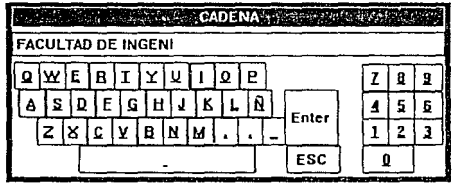
OPERANDO 1	OPERANDO 2	RESULTADO
V	V	V
V	F	F
F	V	V
F	F	F

La quinta columna de botones corresponde a la selección de un valor de dato mismo que se puede definir dependiendo del tipo que se deseé tener. De aquí que tenemos el botón "C" para un valor de tipo carácter, el botón "N" para un valor numérico, el botón F para la definición de un valor de tipo fecha y por último tenemos el botón "B" para definición de un valor booleano. A continuación definimos el procedimiento para realizar la definición de estos valores.

CARACTER.

Para realizar la definición de una cadena de caracteres mediante el botón identificado como "C", procedemos a pulsar el botón correspondiente con lo que obtenemos la siguiente ventana:

VENTANA CARACTER.



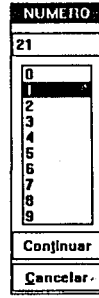
Como se observa, tenemos por una parte el conjunto de caracteres y por otro lado el conjunto de números, además se cuenta con la barra espaciadora y con dos botones mas que corresponden al botón de "ENTER" y al botón de "ESC". En la parte superior de la ventana se encuentra un área misma que se utiliza para la captura de la cadena alfanúmerica. Para la definición de la expresión únicamente hay que pulsar la tecla del carácter deseado ya sea directamente en el teclado de la PC o bien con ayuda del ratón pulsamos el botón del carácter. Esta acción se va registrando en el área de captura con el despliegue del carácter elegido. También cabe señalar que en cualquier momento podemos posicionarnos en cualquier punto de la cadena y realizar operaciones de inserción o borrado.

Una vez que hemos definido la cadena de caracteres podemos pulsar la tecla de ENTER para indicar su aceptación o bien la tecla de ESC para abandonar la operación. En caso de haber aceptado la cadena

definida esta se inserta en la expresión que estamos definiendo como query en la última posición en la que nos encontrábamos.

NUMEROS.

Al pulsar el botón identificado como "N" aparece en pantalla el siguiente despliegue:

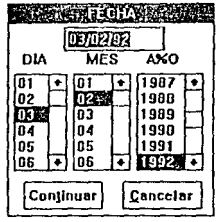


Como podemos observar tenemos en la parte superior el área de captura del dato y bajo esta área tenemos la ventana de selección misma que contiene a todos los dígitos posibles, y por último en la parte inferior tenemos los botones de "Continuar" y de "Cancelar". Ahora bien, el procedimiento para definir un valor numérico es únicamente posicionarnos en el dígito deseado y pulsar el botón izquierdo del ratón, acción que lleva al despliegue del dígito seleccionado en el área de captura. En esta área también tenemos movimiento, de modo que podemos posicionarnos en cualquier punto y realizar operaciones de borrado o inserción de otros dígitos.

Ya una vez definida la expresión numérica procedemos a pulsar el botón de "Continuar" en caso de aceptar el valor o bien el botón de "Cancelar" en el caso de no aceptar la operación seleccionada.

FECHA.

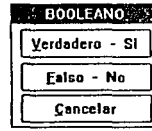
La siguiente ventana corresponde a la selección de un valor de tipo fecha:



Como observamos en esta pantalla tenemos primeramente en la parte superior el área de captura, en la parte central tenemos tres áreas de selección correspondientes al día, mes y año; y por último en la parte inferior de la ventana tenemos los botones de "Continuar" y de "Cancelar". El procedimiento para definir una fecha es el siguiente: para seleccionar el día o el mes o el año según corresponda podemos posicionarnos en el área donde se encuentran los dígitos y manteniendo presionado el botón izquierdo del ratón tenemos un movimiento a través de todos los valores posibles y una vez que quedamos posicionados en el valor deseado podemos seleccionarlo pulsando para ello dos veces el botón izquierdo del ratón. También es posible tener un desplazamiento a través de los valores pulsando las flechas colocadas en la barra correspondiente a cada área. Una vez definido el valor procedemos a pulsar el botón de "Continuar" para aceptar el valor o bien la tecla de "Cancelar" para abandonar la operación.

BOOLEANO.

La selección de un valor de tipo booleano se realiza a través de la siguiente pantalla:



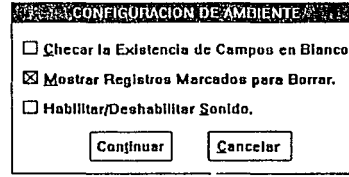
Como se observa tenemos tres botones: los dos primeros permiten definir el valor de verdadero o bien de falso. Y el tercer botón permite cancelar la operación de selección.

IX.6.4 SUBMENU DE UTILERIA.

Este submenú contiene las opciones correspondientes a las utilerías del sistema, como son:

CONFIGURACION.

Esta opción permite la activación o desactivación de parámetros relacionados con el ambiente de ejecución del sistema AVE. Al ser seleccionada se obtiene el siguiente despliegue en pantalla:



Donde observamos que tenemos la posibilidad de activar o desactivar tres parámetros. Para realizar esto, únicamente basta con posicionarnos en el recuadro que antecede al parámetro deseado y pulsamos el botón izquierdo del ratón acción que activará (aparece una X) o desactivará la opción.

Como se mencionó anteriormente se pueden definir tres parámetros, los cuales de describen a continuación. El primer parámetro corresponde al chequeo de la existencia de campos en blanco, esto involucra que en caso de encontrarse activa esta opción en el momento de estar agregando información en los archivos, esta se verifica de tal forma que no es posible introducir un registro en

blanco, en caso contrario se almacenará de no encontrarse activo este parámetro.

El segundo parámetro que se puede definir es el correspondiente a la activación o desactivación de la visualización de los registros marcados para borrarse.

Por último tenemos el parámetro para la activación o desactivación de los efectos de sonido que se manejan durante la ejecución de la aplicación.

IMPORTAR TEXTO.

Esta opción permite traer información que se encuentra almacenada en un archivo de texto y agregarla en un archivo de base de datos del sistema AVE ya existente o bien en un nuevo archivo.

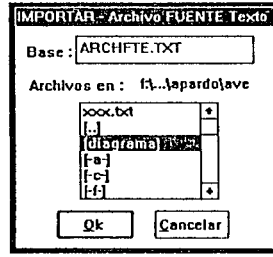
El proceso se realiza de la siguiente manera: primeramente al seleccionar la opción de "Importar texto", aparece un submenú con dos opciones que son :

- "Agregar en un nuevo ABD".
- "Agregar en un ABD ya existente".

De aquí seleccionamos la opción deseada, acción que nos lleva a cada uno de los casos, tal como se indica a continuación.

Agregar en un ABD nuevo.

Al seleccionar esta opción tenemos la siguiente pantalla:

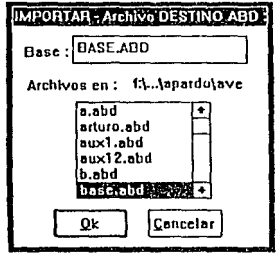


Donde tenemos que especificar el nombre del archivo fuente que almacena la información en texto. Para realizar esta acción tenemos las siguientes posibilidades:

- Unicamente basta con seleccionar el nombre del archivo y pulsar la tecla de "ENTER" o bien el botón de "OK".

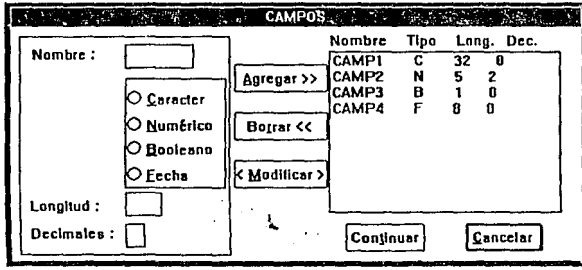
- Seleccionar el nombre, navegando en la ventana que muestra los directorios y nombres de los archivos.

Ya una vez especificado el nombre del archivo se despliega la siguiente pantalla:



En donde especificamos el nombre del archivo en donde se almacenará la información. Dicha especificación puede realizarse de igual manera.

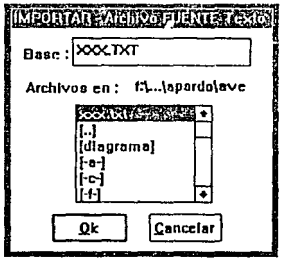
Al especificar este nombre se procede a la definición de la nueva estructura:



Ya una vez definida, se procede a la importación del archivo de texto donde dicho proceso se muestra a través de la siguiente pantalla:

Agregar en un ABD ya existente.

Al seleccionar esta opción tenemos la oportunidad de que la información proveniente de un archivo de texto pueda ser agregada en un archivo de base de datos ya existente. Para ello primeramente hay que seleccionar el nombre del archivo texto, acción que se lleva acabo mediante la siguiente pantalla:



Ya una vez definido, se procede a especificar el nombre de la base de datos en la que se agregará la información, tal y como se muestra en la siguiente pantalla:

Ya definido el archivo destino se procede a realizar la importación, acción que se muestra a través del siguiente desplegado:

IX.6.5 SUBMENU DE OPCION SISTEMA.

Este menú comprende las siguientes opciones:

DOS:

Esta opción permite la ejecución de comandos del Sistema Operativo sin tener la necesidad de salir de la aplicación del DBMS AVE , para ello al seleccionar la opción de DOS después de algunos segundos aparece en pantalla el indicador (prompt) del sistema operativo y nos sitúa en el subdirectorio actual,como lo muestra la siguiente pantalla:

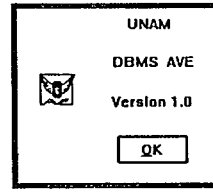
```
Microsoft(R) MS-DOS(R) Version 5.00
 (C)Copyright Microsoft Corp 1981-1991.

F:\US\DESARROL\APARDO\AVE>
```

Como se observa, aparece el indicador o prompt del Sistema Operativo y estando ubicados en este punto podemos realizar cualquier operación propia del sistema, como lo son copias de archivos, borrado desplegado de directorios, etc, una vez finalizadas las operaciones

deseadas podemos retornar a nuestra aplicacion DBMS AVE tecleando el comando "EXIT", acción que inmediatamente nos lleva a la aplicación.

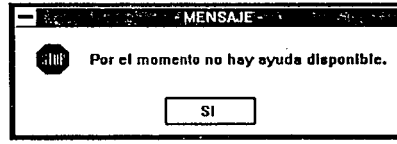
VERSION.

Esta opción permite conocer el nombre y versión de la aplicación, mediante el desplegado de la siguiente pantalla:



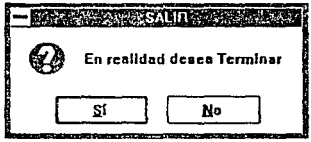
AYUDA:

Mediante esta opción es posible obtener el desplegado de la ayuda correspondiente a la aplicación, cabe señalar que por el momento esta opción no se encuentra habil.tada, por lo que únicamente al ser seleccionada obtenemos la siguiente pantalla:



SALIR.

Esta opción nos permite finalizar la ejecución de la aplicación DBMS AVE, para ello al realizar la selección de esta opción obtenemos la siguiente pantalla:



Como observamos se solicita la confirmación de la operación donde únicamente hay que seleccionar la respuesta adecuada y con ello finalizar la aplicación o bien continuarla.

CAPITULO X

CONCLUSIONES

Con el crear,
es el enseñar la actividad intelectual superior.

X. CONCLUSIONES

X.1. HISTORIA DE LA INTERFACE

El sistema tuvo como premisa el desarrollo de un manejador de Bases de Datos con una interface amistosa hacia el usuario, por lo cual se decidió que dicha interface sería con el uso de ventanas, de esa manera el usuario podría navegar dentro del sistema de una manera sencilla y eficaz. La tecnología existente y apta para el desarrollo en ese momento resultó ser el lenguaje de programación "C" (Turbo C) de Borland Inc. Existían tres métodos posibles para el desarrollo del manejo de ventanas:

i) - Método de Alto Nivel.
ii) - Método de Bajo Nivel.
iii)- Método de escritura directa a la memoria de video.

X.2. METODO DE ALTO NIVEL

La primera técnica (*Alto Nivel*) resultaba de fácil manejo, ya que se basa en el uso de funciones de la interrupción 21H del MS-DOS, se requería unicamente de llenar los registros del sistema (AH, AL, DX, DS, DX, etc) con los datos necesarios para después llamar a la interrupción 21 Hexadecimal. En particular, las funciones utilizadas eran las dedicadas a la salida y entrada hacia el video y estas son :

Hexadecimal	Decimal	Nombre de Función
1	1	Entrada de caracter con eco al video.
2	2	Salida de un caracter al video.
6	6	E/S directa de consola.
8	8	Entrada de un caracter al video sin eco.

Esta forma ofrecía independencia con el hardware usado pero no resultaba ser la mejor opción, ya que a pesar de su facilidad de uso no ofrecía control directo del video. Por ejemplo, en cuanto se enviaba una interrupción, los datos eran escritos en la memoria del video en cualquier momento entonces se presentaba una asincronía entre el barrido de la memoria del video y la escritura a ella, provocando una especie de nieve en la pantalla.

X.3 METODO DE BAJO NIVEL

El segundo método (*Bajo Nivel*) consiste en llamadas directas al BIOS (Basic Input Output System) del equipo a través de algún software, esto provocaba un acceso rápido al controlador del video además de proporcionar primitivas para graficación. Este método resultaba ser bueno solo que el programa obtenido debía correr en cualquier IBM PC o compatible, en caso contrario se provocaba conflictos, es decir, en equipos con MS-DOS no operaba correctamente. Básicamente el controlador de video era accesado a través de la Interrupción 10 H. Este tipo de programación fue usado en algunas funciones dentro del sistema original AVE :

Hexadecimal	Decimal	Función
0	0	Inicializar tipo de video.
1	1	Cambio del tipo de cursor.
2	2	Posicionamiento del cursor.
3	3	Lectura de la posición del cursor.
6	6	Inicialización de ventana o scroll hacia arriba.
7	7	Inicialización de ventana o scroll hacia abajo
8	8	Lectura del atributo del cursor.

Sin embargo no fue usada para la lectura y escritura de caracteres al video, ya que de igual forma que el método anterior provocaba una especie de nieve en el video.

X.4 METODO DE ESCRITURA DIRECTA A LA MEMORIA DE VIDEO

El tercer método resultó ser mas viable, ya que evitaba los problemas indicados. Mediante pequeños programas escritos en lenguaje ensamblador y la liga entre lenguaje "C", se permitía la escritura directa y en sincronía con el barrido a la memoria de video, permitiendo solucionar el problema de incompatibilidad en hardware, ya que el sistema se reajustaba dependiendo del hardware presente. Así pues, se desarrollaron rutinas primitivas en ensamblador para obtener sincronía, escribir y leer caracteres a la memoria de video. Este proceso resultó ser el mejor por su velocidad y por la desaparición de la nieve en pantalla. Básicamente el funcionamiento se apoyaba en técnicas de mapeo hacia la memoria del video dependiendo de la posición ocupada en la pantalla, así pues, la función de mapeo usada resultó ser :

$$offset = ((Ren * 50h + Col) * 2) + (pagina * 1000H)$$

Bajo el lenguaje "C", se desarrollaron rutinas de mas alto nivel, las cuales accesaban a las escritas en lenguaje ensamblador, de esta manera se desarrollaron rutinas tales como :

- Salvado de Regiones
- Escritura de Regiones Salvadas
- Edición de campos de captura
- Scroll horizontal de campos de captura
- Generación de marcos
- Generación de ventanas
- Navegación de menúes
- Control del cursor en pantalla
- Scroll vertical en ventanas entre otras.

Con todas estas funciones se logró obtener una interface agradable a la vista y un fácil manejo del usuario final.

Cuando todo el desarrollo de la interface estaba lista, el desarrollo se suspendió por diversos motivos, lo cual provocó finalmente que todo el desarrollo se viera obsoleto después de varios meses.

X.5 USO DE HERRAMIENTAS DURANTE EL DESARROLLO

El desarrollo lento y largo de un sistema conlleva a tener una serie de problemas difíciles de evadir, tal es el caso del *DBMS AVE*, que durante 2 años se presentaron varias actualizaciones del software utilizado para el desarrollo, basta mencionar que el sistema en un inicio fue desarrollado utilizando el ambiente y compilador de "Turbo C v 1.0" de la compañía Borland Inc., después apareció la nueva versión del compilador "Turbo C v 2.0" que a pesar de ser compatibles no dejan de tener diferencias tanto en el ambiente como en la compilación. Posteriormente surgió el compilador de Microsoft C versión 5.0, y según las tendencias del mercado el sistema DBMS AVE migró a tal ambiente, este paso fue importante y difícil, ya que existían grandes diferencias y no eran reportadas : el sistema compilaba bien pero no realizaba las tareas adecuadas; posteriormente el ambiente Windows 3.0 surgió con un kit de desarrollo (SDK de Microsoft) compilable con Microsoft C 5.0, y que finalmente se migró a Microsoft C 6.0.

En esta última fase surgieron una gran cantidad de herramientas para el desarrollo como depuradores (Quick C de Microsoft), herramientas CASE (CASEW), nuevos ambientes de programación para Windows 3.0 (Quick C y C++ de la compañía Sortheck), nuevos lenguajes como Basic en el sistema Visual Basic de Microsoft, pero debido al avance del desarrollo y al poco beneficio obtenido se decidió finalmente no migrar hacia estos ambientes.

X.6 OPORTUNIDAD DE MERCADO

El avance tecnológico ha permitido el desarrollo de nuevos ambientes de programación con una tendencia cada vez mas orientada a obtener una interface amistosa hacia el usuario final, tal es el caso de Windows 3.0, el cual proporciona el manejo de ventanas con una interface gráfica, pudiendo trabajar con un ambiente de multitareas. El auge de Windows 3.0 ha sido tal, que la mayoría de los sistemas ya existentes bajo una interface texto, ahora están migrando hacia una interface gráfica (GDI de Windows) y ahora se puede obtener un mismo sistema bajo los dos ambientes. Durante el desarrollo de DBMS AVE, se presentaron varias tendencias : programación en lenguaje C, programación con desarrollo de ventanas, programación con ventanas implícitas, multitareas bajo DOS y finalmente bajo Windows 3.0; por tal motivo y viendo las tendencias del mercado, se decidió migrar el código hasta entonces desarrollado y por desarrollar hacia dicho ambiente. El proceso requería del uso de nuevo hardware y software, además de reestructurar la mayoría de los procesos los cuales contenían entradas basadas en las librerías estándar de C (printf, scanf, getch, putchar, puts, gets, etc.). Windows, en cambio ofrece una manera totalmente diferente e incompatible con el concepto hasta entonces conocido. Cinco meses y medio fueron requeridos para la recopilación de Hardware y Software, además de estudio para finalmente familiarizarse con el ambiente de programación.

X.7 QUE ES AVE ?

Hasta este momento, nuestra atención ha estado centrada sobre el diseño y manejo de AVE, probablemente sea necesario aclarar acerca de su especie pues, como en todos los mundos conocidos por el hombre se hace necesario clasificar y cualificar con regular precisión a sus entes respectivos.

Muy al inicio de este escrito, nos referimos a nuestro producto como un manejador de archivos, entonces, ¿porqué clarificar al respecto?. Bueno, no es que tengamos remordimiento alguno respecto al calificativo imputado, sino que en ese lugar solo pretendimos situarlo, es decir, demarcar su entorno en el manejo de datos y su alcance, a modo de no crear exageradas y/o falsas expectativas. Ahora, una vez observado y valorado sus posibilidades y limitaciones, solo se pretende dar un recorte mas próximo a lo que pudiera ser.

Entonces cabe la pregunta: ¿en qué estado de la evolución softwareiana se haya AVE?.

Pensamos que en ninguno de manera rigurosa pues, por un lado presenta en cierta medida capacidades atribuibles a un sistema manejador de archivos; por el otro, creemos que conceptualmente va un poco mas allá ya que, se observan facciones características de un sistema manejador de bases de datos, aunque en cierta medida disminuidas.

BIBLIOGRAFIA

... las vendas cayeron de mis ojos,
las dudas se desvanecieron,
y una sensacion de tranquila certidumbre ocupó su lugar.

BIBLIOGRAFÍA

Microsoft Windows Guide to Programming
New for Version 3
Microsoft Corporation

Microsoft Windows Programming Tools
New for Version 3
Microsoft Corporation

Microsoft Windows Programmer's Reference
New for Version 3
Microsoft Corporation

Microsoft C 6 Reference
For MS OS/2 and MS-DOS Operating Systems
Microsoft Corporation

Microsoft C 6 Run-Time Library Reference
For MS OS/2 and MS-DOS Operating Systems
Microsoft Corporation

Advanced MSDOS
The Microsoft Guide for Assembly Language and C Programmenrs.
Microsoft PRESS

Using Turbo C
Herbert Schildt
Borland Osborne / McGraw-Hill
Programmin Series

Turbo C, User's Guide
Version 2.0
Borland International

Advanced Turbo C
Herbert Schildt
Borland-Osborne / McGraw-Hill
Programming Series

The Waite Group's Microsoft C Bible
Nabajyoti Barkati
Howard W. Sams & Company

File Structures, a Conceptual Toolkit
Bill Zoellick
Michael J. Folk
Edit. Addison Wesley

Turbo Toolbox Reference Manual
Borland Inc.