UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Facultad de Ingeniería

DESARROLLO DE CODIGOS FUENTES DE UN MANEJADOR DE BASES DE DATOS.

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A :

JOAQUIN TOMAS MEROÑO

Director M. en C. Ricardo Ciria Merce

Ciudad Universitaria

México, D. F. 1986

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

T E M A R I O

TEMA IV ELEMENTOS ESTRUCTURALES DE LA BASE DE DATOS

TEMA V RUTINAS PRINCIPALES DEL MANEJADOR DE BASES DE DATOS

TEMA I INTRODUCCION

Así como la ciencia de la computación ha avanzado, también los métodos de almcenamiento de datos se han vuelto más y más sofisticados. El manejo de almacenamiento de datos ha evolucionado de simples archivos secuenciales, a archivos de acceso aleatorio, organizados para integrar sistemas de bases de datos. Un manejador de bases de datos ofrece un aumento de las capacidades del usuario, pero por la naturleza de las características agregadas, más planeación es requerida en el desarrollo de los sistemas, para una operación óptima. El tiempo que se invierte en la planeación de la base de datos es justificado después, una vez que esta ha sido establecida.

I.1 ¿QUE ES UNA BASE DE DATOS ?

Una base de datos es una colección de archivos lógicamente relacionados, los cuales contienen datos e información estructural. Los apuntadores en una base de datos permiten al usuario accesar información relacionada y dirigirse a través de los archivos de datos. La organización de una base de datos puede adquirir diferentes formas; dos ejemplos de ésto serían la estructura jerárquica y la estructura de red.

La estructura jerárquica ha sido un crecimiento natural a partir de las primeras técnicas de organizción de archivos. Los datos deben ser accesados a través de niveles de calificadores. Por ejemplo en una base de inventarios, para poder accesar la información de un producto habría que accesar primero en que almacén y a que lote pertenece. Se utilizan archivos de referencia y de enlace para una asociación lógica y lograr accesibilidad. Cuando el número de archivos de datos crece y su interrelación con ellos, se convierte una estructura más compleja, los requerimientos de archivos de referencia y de enlace crecen en forma exponencial por el requerimiento de ligas. Esto da como resultado demasiado trabajo para accesar datos.

Las bases de datos estructuradas bajo una red funcionan

en base a la premisa de que solo cuando un grupo de datos (Archivo de datos) está relacionado con otro grupo, debe haber una liga directa entre ellos, de ésta manera los archivos de referencia y de enlace no se requieren más. El incremento de la complejidad de la base depende directamente del número de relaciones directas existentes entre un cierto número de archivos de datos, cuando los archivos de datos se consideran como nodos con ligas de acceso directo conectándolos se ha formado una base de datos de estructura en red.

En una base de inventarios cada almacén sería un archivo diferente, con sus productos.

I.2 OBJETIVOS DE UN MANEJADOR DE BASES DE DATOS.

Un método efectivo de organización de bases de datos implica también el mantenimiento de esta. Sin embargo hay muchas razones prácticas para considerar un manejador de bases de datos como la solución a ciertos problemás de almacenamiento de datos.

Los objetivos de un manejador de base de datos son la seguridad y la integridad de estos, el manejo adecuado, inmunidad a cambios externos, y facilidad de uso en diferentes medios. La independencia de la base de datos es lo más importante . A partir de que los datos de la base son centralizados, se vuelve más sencillo concentrar el control de ellos. Esta centralización nos da una independencia de la base de datos y los programas de aplicación. Las modificaciones a los programas de aplicación no afectarán a la base de datos y las modificaciones a la base de datos no afectarán a los programas de aplicación. Esta independencia provee los fundamentos para un sistema de información dinámico y con demandas de crecimiento. Un manejador de bases de datos debe ser orientado al usuario, de tal forma que este aprenda fácilmente, entendiendo y así creando una amplia gama de aplicaciones. El usuario debe ser provisto de ayuda para la seguridad y el desarrollo de sus sistemas.

I.3 ¿ POR QUE USAR UNA BASE DE DATOS ?

El principal beneficio por el cual utilizar un manejador de bases de datos es el ahorro de tiempo . Este ahorro se puede manifestar de las siguientes formas.

+ CONSOLIDACION DE ARCHIVOS.

La mayoría de los procesos de información que se utilizan en diferentes areas de aplicación contienen datos repetidos. Por ejemplo el nombre de una materia prima puede aparecer en un archivo de inventario, en un archivo de fórmulas o en un archivo de tránsito. Los datos entre éstos tres archivos tal vez varíen ligeramente causando de esta manera un desperdicio de memoria secundaria. La información redundante se vuelve un gran problema en sistemas de información grandes.

La consolidación de archivos dentro de una base de datos elimina la mayoría de las redundancias. Con el uso de apuntadores los registros de información relacionados lógicamente son ligados aunque estos esten físicamente separados. En el caso del ejemplo anterior solo un archivo existiría y cada programa accesaría la información que requiriera. Al existir solo un registro para accesar y modificar se ahorra tiempo de acceso y espacio en memoria.

+ INDEPENDENCIA DE PROGRAMAS

Las estructuras convencionales de archivos tienden a ser rígidas e inflexibles. La naturaleza de los métodos de archivos convencionales requerían que la lógica de los programas de aplicación estuviera estrechamente ligada al diseño de los archivos. Cuando es necesario alterar la estructura de un archivo, un programa debe ser escrito para cambiar el archivo, y los programas que lo accesaban deben ser modificados de tal manera que reflejen el cambio realizado en dicho archivo. Es bien conocido que en sistemas una gran parte del tiempo se

utiliza en el mantenimiento de programas.

Este manejador de bases de datos permitirá que la estructura de la base de datos sea independiente a los programas de aplicación. La relación entre registros será definida independientemente. Los cambios en la base de datos deberán ser incorporados únicamente en aquellos programas que manipulan el dato que cambio, los programas solo accesarán la porción de la base que sea requerida por su aplicación lo que evita que éstos tengan que ser modificados sin accesar el dato que cambió.

± VERSATILIDAD

Las técnicas de organización de archivos convencionales permiten un acceso limitado a los datos que contienen. La mayoría de las estructuras permiten un acceso de una sola llave, con un acceso relacional adicional, permitido solamente a través de la implementación de un programa de soporte de gran nivel.

Este manejador permitirá el acceso con varias llaves así como a través de una variedad de métodos.

± SEGURIDAD DE DATOS

Los manejadores convencionales contienen una seguridad para los datos muy limitada. El acceso a un dato puede ser negado al usuario con el acceso al sistema solo por la provisión de un medio físico.

Este manejador proveerá seguridad al archivo .

± DESARROLLO DE PROGRAMAS

La estructura de la base de datos puede ser definida y construida sin el uso de programas de aplicación especiales. Partiendo de que el control de las ligas de la base de datos estará bajo el control del software del manejador, el programador no necesitará preocuparse por probar la estruc

tura y podrá concentrarse en la programación de la aplicación.

± MANTENIMIENTO DE PROGRAMAS

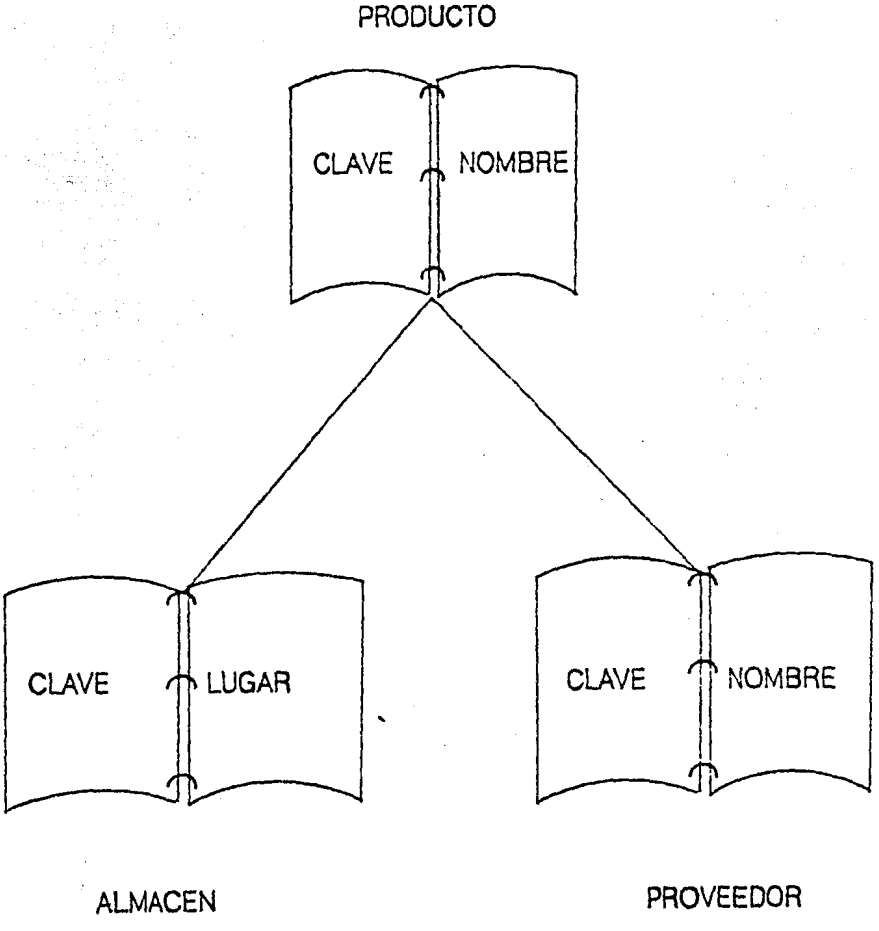
Durante la vida de un sistema, los procesos requeridos definen el uso de los datos . Así como la organización de los archivos varía con las necesidades de la aplicacion, algunos cambios pueden ser requeridos con un pequeño impacto sobre los programas de aplicación existentes. Los cambios en la estructura de una base de datos existente afectarán solo a aquellos programas que procesan los datos cambiados;ningún otro programa debe ser recomplilado para reflejar el cambio de estructura de la base.

Sumarizando: el uso de un manejador de bases de datos puede eliminar gran parte del tiempo en programacion, el cual se puede aplicar a otras tareas como diseño y producción.

TEMA II CARACTERISTICAS DE LA BASE DE DATOS.

II.1 RELACION ENTRE ARCHIVOS DE DATOS

La relación directa entre diferentes grupos de datos es el fundamento de éste manejador de bases de datos. Cuando uno examina las aplicaciones diarias, se pueden encontrar muchos ejemplos de grupos de datos relacionados con otro grupo de datos y esté a su vez con otro grupo de datos. Por ejemplo, en el caso de los inventarios un producto puede estar relacionado con diferentes almacenes y éstos a su vez con diferentes proveedores. Podemos observar de ésta manera la necesidad de relacionar los archivos para saber así cuantos almacenes pueden solicitar nuestro producto o cuantos proveedores lo puedan surtir.


PRODUCTO
CLAVE
NOMBRE
CLAVE
LUGAR
CLAVE
NOMBRE
ALMACEN
PROVEEDOR

Este tipo de relaciones podrán ir creciendo y variando de acuerdo a nuestras necesidades.

II.2 ESTRUCTURA DE UNA BASE DE DATOS.

Después de haber visto algunas de las características del manejador de la base de datos podremos dar algunas definiciones de los elementos que la conformarán.

II.2.1 BASE DE DATOS.

Una base de datos consistirá en uno o más archivos los cuales tendrán una relación lógica con los otros, un archivo a su vez consistirá en uno o más registros de longitud fija y cada registro será formado por uno o más campos.

II.2.2 CAMPO.

Un campo es el dato accesible más pequeño dentro de la base. Cada campo consiste de un valor referenciado por el nombre del campo, típicamente seleccionado para describir el valor del dato. En resumen, el nombre de un campo puede referenciar diferentes valores del mismo, cada uno de éstos valores están a su vez asociados a un registro diferente.

II.2.3 TIPOS DE DATOS

El diseñador de la base definirá cada campo como un tipo particular, dependiendo de que tipo de información va a ser almcenada en el campo, puede ser entero, real etc. . .

II.2.4 REGISTROS DE DATOS

Un registro es un conjunto ordenado de campos, el orden de los campos en el registro es definido cuando la base de datos es creada los registros deberán ser definidos con un máximo límite de campos ninguno de los cuales deberá ser repetido dentro del registro. La longitud del registro será la suma de las longitudes de los campos que lo forman, cuando la información del registro sea

almacenada en la base, se le agregará información estructural adicional para su manejo.

II.2.5 ARCHIVOS DE DATOS.

Un archivo de datos es una colección de registros, llamados ocurrencias, donde cada registro contiene valores diferentes para un mismo campo. Por ejemplo, un archivo de datos conteniendo ocurrencias de productos puede tener datos como nombre, código, lote etc. Cada archivo es referenciado por un nombre único y se almacena en un disco. Cuando un archivo es creado por la definición de la base de datos, debe haber sido especificada la capacidad de éste en registros. Cada registro tendrá un número de registro asignado, que será utilizado por el manejador para el acceso a este. Este número se llama número relativo de registro. Cuando la capacidad del archivo se ha especificado, el archivo es creado para ese número de registros, sí un registro en un número relativo no tiene información se dice que es un registro vacío.

II.2.5.1 ARCHIVOS MAESTROS

Un archivo Maestro tiene las siguientes características.

+ Se utilizan para guardar información relacionada a un identificador único.

+ Permiten rápido acceso a la información, pues uno de los campos del registro, llamado campo llave, determina la localidad del mismo dentro del archivo.

+ Pueden estar relacionados a archivos detallados por medio de campos llaves similares, los cuales sirven como índice al archivo de detalle.

II.2.5.2 ARCHIVOS DETALLADOS

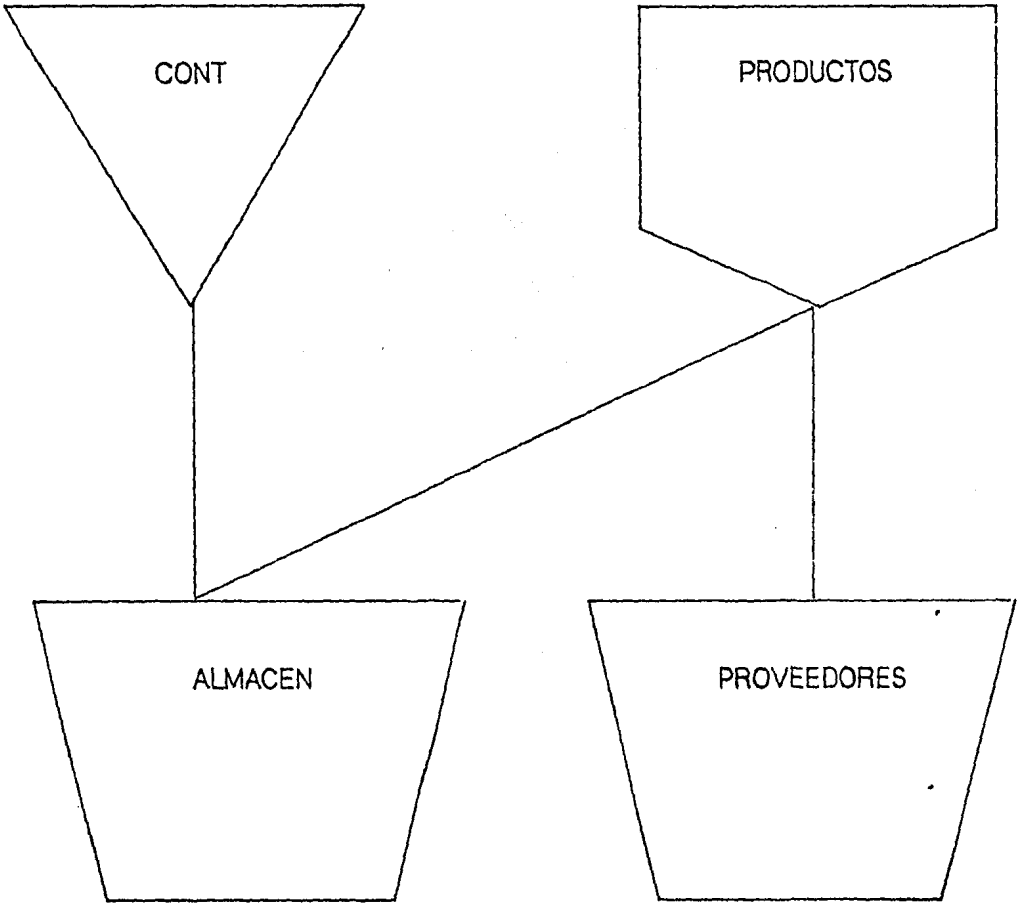
Los archivos detallados se caracterizan por lo siguiente:

+ Se utilizan para almacenar información sobre hechos relacionados; Por ejemplo la información de todos los pedidos relacionados a un solo producto.

+ Nos permiten accesar toda una serie de información relacionada a un ente único.

La localidad de memoria en un archivo de detalle no tiene relación con su contenido. Cuando se desea agregar nueva información a un archivo de detalle, se coloca en la primera localidad disponible.

ARCHIVOS MAESTROS



ARCHIVOS DE DETALLE

A diferencia de un archivo maestro que contiene solo un campo llave, los archivos detallados pueden contener varios, el valor de un campo llave en un archivo de detalle no necesita ser único y puede haber varios registros con el mismo valor en el campo llave.

El manejador guardará información en forma de apuntadores, que enlazarán todas las ocurrencias con el mismo valor de campo llave, de ésta manera se formará una cadena.

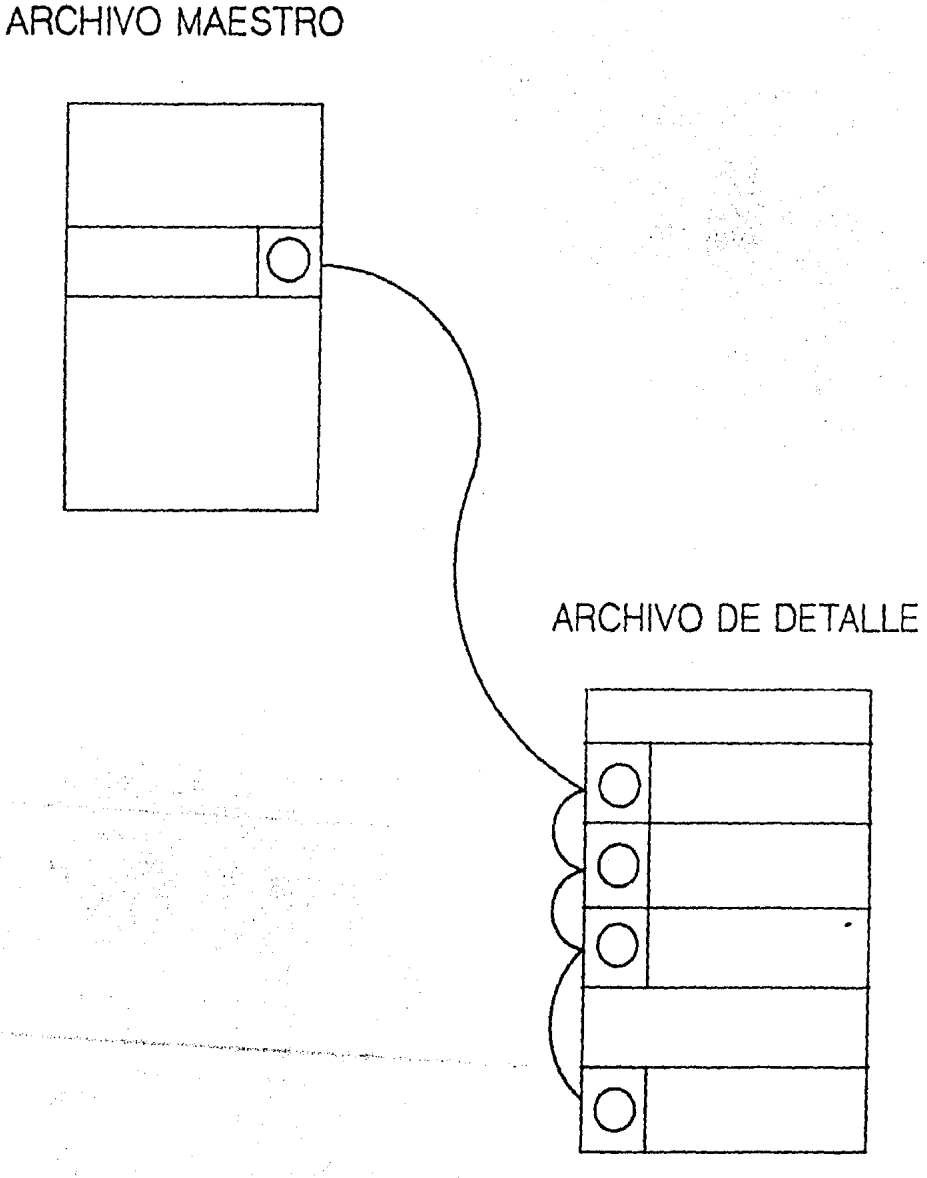
II.2.5.3 LIGAS

Un archivo maestro debe tener un solo campo llave que puede estar relacionado con uno o más archivos detallados. Los campos llaves en el maestro y en el detallado deben ser del mismo tipo y tamaño, ésta relación formará una liga. Un archivo maestro puede formar ligas con varios archivos detallados.

Un archivo de detalle puede tener registros con varios campos llaves que lo relacionen a su vez con varios archivos maestros.

para cada liga de un archivo maestro, se formará una cadena con cada campo llave, que estará formada por varios registros en un archivo de detalle con el mismo valor de campo llave del campo llave que el maestro relacionado. la información del archivo maestro contiene un apuntador a la cadena. Esta información se llama cabeza de la cadena.

Dentro de un archivo de detalle, todas las ocurrencias que comparten el valor de un campo llave son juntadas en una cadena. Una cadena puede contener cero ocurrencias o tantas como haya espacio en el archivo.


ARCHIVO MAESTRO
ARCHIVO DE DETALLE

II.2.5.4 ARCHIVOS MAESTROS MANUALES Y AUTOMATICOS.

Un archivo maestro puede ser manual o automático, estos dos tipos de archivos tienen las siguientes caracteristicas.

MANUAL	AUTOMATICO
Puede permanecer solo, no necesita estar relacionado con algún archivo detallado.	Debe estar relacionado a uno o más archivos detallados.
Puede contener otros campos en el registro además del campo llave.	Los registros solo contienen al campo llave.
Las ocurrencias deben ser agregadas o borradas explícitamente. Una ocurrencia relacionada en un detallado no puede ser agregada hasta que exista un campo llave en el maestro. Cuando la última ocurrencia dentro de un detallado relacionado es borrada, la ocurrencia del maestro permanece. Antes de que se borre una ocurrencia en el maestro, deben haber sido borradas todas las ocurrencias en el detallado.	El manejador agrega o borra ocurrencias cuando se necesita, en base a la alta o baja de datos en el detallado relacionado. Cuando en un detallado relacionado se agrega una ocurrencia con un valor de campo llave no existente en el maestro, se da de alta una ocurrencia automáticamente. Si se borran todas las ocurencias de un detallado relacionado la ocurrencia en el maestro relacionado se borra automáticamente.
Los campos llaves existentes en un maestro sirven como una tabla para campos llave legítimos, para todos los archivos detallados relacionados.	

Los diseñadores de la base pueden utilizar:

+ Archivos maestros para asegurarse de que un campo llave dado de alta en un archivo de detalle es válido.

+ Archivos automáticos para ahorrar tiempo cuando los valores de un campo llave son impredecibles o son tan numerosos que es deseable evitar el trabajo de dar de alta y de baja en el maestro

Un archivo maestro manual puede ser definido en la generación de la base, con cero ligas a detallados y de esa forma puede funcionar como un tipo especial de archivo detallado.

1.- Uno organizado aleatoriamente.

2.- Uno que no está ligado a ningún archivo maestro.

3.- Uno en el cual solo hay una ocurrencia para un campo llave.

II.2.6 ARCHIVOS PROPIOS DEL MANEJADOR DE BASES DE DATOS

Todos los elementos de la base son almacenados en archivos de disco. Estos son creados cuando el diseñador corre el generador de bases de datos. Las localidades de disco en donde éstos archivos serán creados han sido especificadas en el esquema para el generador. El código de seguridad de todos los archivos que se creen será igual al código de seguridad de la base de datos. Los archivos de datos de la base serán del tipo binario y no otro para que no puedan ser leídos más que por las rutinas de la base de datos.

II.2.6.1 ARCHIVO RAIZ

Es creado en la generación de la base con un nombre idéntico al de la base de datos. Sirve como un punto común de acceso a información fuente de la base de datos.

II.2.6.1 ARCHIVOS DE DATOS

Existe un archivo de datos para cada archivo de la base de datos. En la creación de la base el tamaño y número

de registros dentro de un archivo, es determinado por la información del archivo Raíz. Los archivos de datos son creados en la generación de la base inmediatamente después del archivo Raíz.

Cada archivo de datos tiene el mismo código de seguridad que el archivo Raíz. El nombre de cada archivo de datos es igual al archivo de la base que representa.

El tamaño de cada archivo de datos será determinado por el tipo de archivo de la base que contendrá, es decir por su tamaño de registros, ligas y capacidad en número de registros. Cuando es creado se hace con la capacidad definida sin importar cuantos registros contendrá originalmente. Los registros que no tengan informacion se consideran registros vacíos.

II.3 COMO DESARROLLAR UNA BASE DE DATOS

Cuando se va a desarrollar una base de datos se deben responder las siguientes preguntas.

1.-¿Quién utilizará la base de datos? En muchas ocasiones una base de datos es diseñada y soportada por un grupo, para que otro grupo la utilice, las necesidades de todos los miembros del segundo grupo deben ser consideradas antes de diseñar la base para asegurar de esta manera que éstas necesidades pueden ser cumplidas de la forma más eficiente. Se debe saber la información a la que cada usuario tendrá derecho antes de crear la base.

2.-¿Cómo serán los campos y los registros ? En éste punto se analizarán las necesidades de campos de los diferentes usuarios para resolver los posibles conflictos entre ellos de la mejor manera posible.

3.-¿Cuáles serán los campos llave ? Ya que han sido definidos los campos se verán las relaciones entre archivos mediante los campos llave para definir todos los que se necesite.

TEMA III DEFINICION DE LA BASE DE DATOS.

III.1 CONVENCIONES DEL LENGUAJE DE DEFINICION DE DATOS.

El esquema o definición de la base de datos se hará con la ayuda del programa GENBA (Generador de la Base de Datos). Este programa es un del tipo interactivo con el cual sostendremos un diálogo para definir las características de nuestras bases de datos.

III.2 ESTRUCTURA DEL ESQUEMA DE LA BASE DE DATOS.

El generador de la Base consta de cuatro partes principales :

+ Definición de la base de datos.
+ Definición de los archivos.
+ Desplegado de Archivos.
+ Definición de Campos.
+ Generación de Archivos en disco.

Las cuales son parte del diálogo de definición y se irán presentando en el orden mencionado al ejecutar al programa GENBA.

III.2.1 DEFINICION DE LA BASE DE DATOS.

Esta etapa del generador se encargará de recopilar toda la información necesaria para formar la tabla de Control de la base de datos, que es una de las partes importantes del archivo Raíz de la Base de Datos. Cualquier Archivo Raíz contará solamente con una tabla de control, la cual es única para cada Base de Datos.

Las preguntas que se recibirán por parte del programa GENBA en ésta etapa son las siguientes :

>>> INGRESA EL NOMBRE DE LA BASE DE DATOS (6 CARACTERES) ? _

El nombre de la Base de datos deberá teclearse con 6 caracteres alfabéticos, cualquier otra respuesta será rechazada.

>>> INGRESA CODIGO DE SEGURIDAD DE LA BASE DE DATOS (2 CARACTERES)

El código de seguridad deberá ser de dos caracteres alfabéticos, cualquier otra cosa será rechazada. Este código dará protección a los archivos para que nadie pueda purgarlos ni escribir en ellos, a menos que lo conozca.

>>> INGRESA LA UNIDAD DONDE RESIDIRA LA BASE DE DATOS (2 DIGITOS) ?

La unidad nos dirá en que porción del disco residirán los archivos de la base de datos y deberá ser de dos dígitos, cualquier otra cosa será rechazada.

>>> CUANTOS CAMPOS TENDRA LA BASE DE DATOS ? _

En ésta pregunta se tendrá que contestar el número total de campos con que contará la base de datos.

>>> CUANTOS ARCHIVOS TENDRA LA BASE DE DATOS ? _

La respuesta deberá ser el número total de archivos que tendrá nuestra base de Datos.

III.2.2 DEFINICION DE ARCHIVOS.

En ésta Parte del generador se contestarán preguntas para crear la tabla de archivos de la Base de Datos, esta tabla es la segunda parte del Archivo Raíz y tendrá una entrada por cada archivo de la Base de Datos.

El programa GENBA nos desplegará las siguientes preguntas tantas veces como archivos se hayan declarado en la etapa de Definición de la Base de Datos.

>>> INGRESA NOMBRE DE ARCHIVO (6 CARACTERES) ?

El nombre del Archivo deberá ser de 6 caracteres alfabéticos, cualquier otra respuesta será rechazada.

>>> INGRESA TIPO DEL ARCHIVO (2 CARACTERES) ? _

Se podrá contestar una de tres opciones :

1) 'MM' : Sí se trata de un archivo Maestro Manual.
2) 'MA' : Sí se trata de un archivo Maestro automático.
3) 'DE' : Sí se trata de un archivo de Detalle.

>>> INGRESA CAPACIDAD DEL ARCHIVO ? _

Se deberá responder el número de registros máximo que tendrá ese archivo.

III.2.4 DESPLEGADO DE ARCHIVOS.

En ésta etapa se despliega la información de todos los archivos de la Base y su número asignado por el generador, para verificar los datos.

ARCHIVO (Nombre del archivo)
TIPO (Tipo del archivo)
CAPACIDAD EN NUMERO DE REGISTRO (nn)
NUMERO DE ARCHIVO ASIGNADO (n)
Teclea Return para continuar >>>>

III.2.5 DEFINICION DE CAMPOS.

En ésta etapa continua el diálogo con el generador, pidiendo los datos necesarios para formar la Tabla de Campos, la cual es la tercera y última parte del Archivo Raíz. Esta tabla cuenta con una entrada por cada Campo de la base de datos.

Se pregunta la información de campos por Archivo.

Archivo (Nombre del archivo) Tipo(xx) Número (n)
Tienes(n) campos disponibles

>>> INGRESA EL NOMBRE DEL CAMPO (6 CARACTERES) (FIN ? _

Sí se responde 'FI' se termina la información de datos para el archivo correspondiente y se pedirá la información de datos para el siguiente archivo.

>>> INGRESA TIPO DE CAMPO ? _

Se responderá el tipo de campo.

1) 'RE' : Sí se trata de un campo real.

2) 'EN': Sí se trata de un campo Entero.

3) 'EN': Sí se trata de un campo Alfanumérico.

Sí el campo es del tipo 'AL' se pedirá otro dato que es el número de bytes que tiene éste campo.

>>> CUANTAS PALABRAS OCUPARA ESTE CAMPO ? _

Se deberá responder de acuerdo al tamaño que se desee.

La siguiente pregunta dependerá del tipo de archivo con el que se éste trabajando.

Sí se trata de uno Maestro.

>>> CUANTAS LIGAS TIENE ESTE CAMPO ? _

Y se responderá :

0 : para cuando no se trata de un campo llave.

n : para cuando se trata de un campo llave. Lo que nos definirá con cuantos archivos de detalle se ligará este maestro.

Sí se trata de un archivo de detalle.

>>> NUMERO DE ARCHIVO CON EL QUE LIGA ? _

Y se responderá :

0 : si no es campo llave.

n : si es llave y ligará con el archivo maestro n.

De ésta forma se definen las ligas de nuestra base de datos. Uniendo las tres etapas vistas se posee toda la información para crear el archivo Raíz, y en base a el crear los archivos de datos de la Base de un tamaño adecuado.

III.2.6 ETAPA FINAL.

En ésta etapa se desplegarán mensajes diciendo el nombre de cada uno de los archivos de datos, su tipo y capacidad, y los datos del archivo Raíz. Con un mensaje diciendo 'creados' si fue exitoso el diálogo de generación.

III.3 ARCHIVO RAIZ.

El archivo Raíz es parte del resultado del generador de la Base de datos. Este archivo es cargado en memoria cuando los programas de aplicación corren, para verificar el manejo de la Base de Datos.

Este archivo consta de tres partes importantes

1) Tabla de control.

2) Tabla de Archivos.

3) Tabla de Campos.

III.3.1 TABLA DE CONTROL.

Esta tabla se conforma con la siguiente información.

1) Palabra 1 Letras 1 y 2 del nombre de la Base de datos.

2) Palabra 2 Letras 3 y 4 del nombre de la Base de datos.

3) Palabra 3 Letras 5 y 6 del nombre de la Base de datos.

4) Palabra 4 Código de seguridad de la Base de Datos.

5) Palabra 5 Unidad de Disco donde reside la Base de Datos.

6) Palabra 6 Número de Campos de la Base de Datos.

7) Palabra 7 Apuntador a la Tabla de Campos.

8) Palabra 8 Número de Archivos de la Base de Datos.

9) Palabra 9 Apuntador a la tabla de Archivos.

10) Palabra 10 Indicador de Base abierta o Cerrada.

11) Palabra 11 Ultimo Registro leído.

12) Palabra 12 Libre en caso de crecimiento del paquete.

13) Palabra 13 Libre en caso de crecimiento del paquete.

14) Palabra 14 Libre en caso de crecimiento del paquete.

III.3.1 TABLA DE ARCHIVOS.

Esta tabla se conforma con la siguiente información.

1) Palabra 1 Letras 1 y 2 del nombre del Archivo.

2) Palabra 2 Letras 3 y 4 del nombre del Archivo.

3) Palabra 3 Letras 5 y 6 del nombre del Archivo.

4) Palabra 4 Unidad de disco donde res de el Archivo.

5) Palabra 5 Tipo de Archivo de Datos.

6) Palabra 6 tamaño del registro del Archivo.

7) Palabra 7 Capacidad del Archivo en Registros.

8) Palabra 8 Número del Campo llave del Archivo.

9) Palabra 9 Número de Archivo Dentro de la Base.

10) Palabra 10 Ultimo registro accesado.

11) Palabra 11 Siguiente registro en la cadena.

12) Palabra 12 Número de registros libres.

13) Palabra 13 Siguiente registro Libre.

14) Palabra 14 Libre para futuro crecimiento.

III.3.1 TABLA DE CAMPOS.

Esta tabla se conforma con la siguiente información.

1) Palabra 1 Letras 1 y 2 del nombre del Campo.

2) Palabra 2 Letras 3 y 4 del nombre del Campo.

3) Palabra 3 Letras 5 y 6 del nombre del Campo.

4) Palabra 4 Tipo de Campo.

5) Palabra 5 Libre para futuro crecimiento.

6) Palabra 6 Libre para futuro crecimiento.

7) Palabra 7 Número del Archivo al que pertenece.

8) Palabra 8 Número de Ocurrencias.

9) Palabra 9 tamaño del campo.

10) Palabra 10 Número de Campo Asignado.

11) Palabra 11 Número de ligas que tiene.

12) Palabra 12 Número de Archivo con el que liga.

13) Palabra 13 Libre para futuro crecimiento.

14) Palabra 14 Libre para futuro crecimiento.

III.4 OBTENCION DEL ARCHIVO DE DATOS.

Teniendo el Archivo Raíz y toda la información se puede calcular el tamaño del registro de cada archivo, y con las capacidades de éstos crear archivos binarios suficientemente grandes para albergar los Datos de la Base y su información de Control, la cual va anexa en cada registro de datos que se verá posteriormente.

Para éstos fines existe una etapa en el generador GENBA donde se crean los archivos de datos con los nombres obtenidos de la Tabla de archivos, reservando espacio en ellos para ser llenado con Datos apoyándonos con las rutinas de La Base de Datos y nuestros programas de Aplicación.

III.5 GENERADOR DE LA BASE EN MODO BATCH.

Por tratarse el programa GENBA de un programa en Pascal, se pensó la posibilidad de crear uno parecido que no leyerá las respuestas de la pantalla, sino de un Archivo de Datos, el cual las tuviera secuencialmente y se creó el programa GENBT.

TEMA IV ELEMENTOS ESTRUCTURALES DE LA BASE DE DATOS

El propósito de éste capítulo es el de proveer información estructural sobre la base de datos para entender mejor el funcionamiento de esta. Se deberá tener cuidado de que los archivos de la base no sean accesados mas que por el manejador de esta, de tal forma que se conserve la integridad de la información.

IV.1 ELEMENTOS ESTRUCTURALES DE LA BASE DE DATOS.

Una base de datos de éste tipo consiste de archivos de datos, mas un archivo conteniendo una descripción de la base de datos llamado archivo Raíz. Además de la información del archivo Raíz en cada registro de los archivos de datos. La información almacenada en cada registro será llamada medio del registro y el propósito de éste será el de informar si el registro contiene datos o no, y si tiene apuntadores a una cadena o no. Un apuntador es un segmento del medio del registro que contiene un número de registro relativo a otro registro en el archivo. Cada registro de datos contendrá un par de apuntadores, uno apuntando al predecesor y otro al sucesor de la cadena. El primer registro en una cadena tendrá su número de registro en el apuntador al registro anterior, el último registro de una cadena contendrá su número de registro en el apuntador al registro posterior.

Esto se verá en posteriores figuras donde se muestran los tipos de registro (para archivos maestros y de detalle).

IV.2 ARCHIVOS DE DETALLE.

Los archivos de detalle pueden pertenecer a una de dos cadenas: la cadena de registros libres, o la cadena de datos. Todos los registros libres no utilizados en un archivo de detalle son encadenados por apuntadores posteriores únicamente, de tal manera que cuando un registro de datos vaya a ser agregado al archivo, ocupará un registro libre rápidamente.

Las cadenas de datos son cadenas de los archivos de detalle que muestran el valor para un campo llave. Cada campo de datos del detallado es contenido en tantas cadenas como llaves haya en ese archivo de datos, el prin

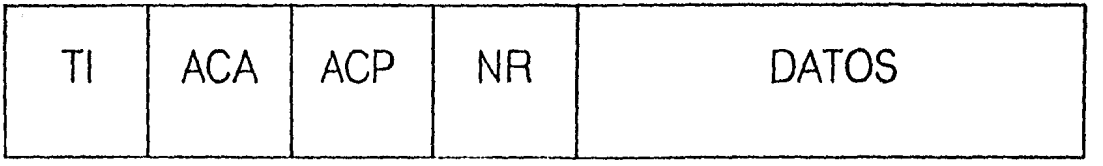
cipio de una cadena será localizado accesando el registro del maestro que tenga también esa llave de acceso. La técnica para realizarlo será explicada a continuación.

La forma del medio del registro es mostrada en la siguiente figura.

El campo marca es cero si el registro está vacío y uno si el registro contiene datos. Si el registro esta vacío, las siguientes dos palabras serán el apuntador al registro libre anterior y el apuntador al registro libre posterior respectivamente. Si la marca es uno, las siguientes 2 palabras son los apuntadores anterior y posterior respectivamente a la cadena de datos, en la siguiente palabra se tiene el número de registro correspondiente. El tamaño de un registro de datos detallado es calculado de la siguiente forma: 4+m palabras, donde m es el tamaño en palabras de los datos que residirán en ese registro.

Los registros libres se ligaron doblemente, pues de esa manera se puede asegurar una forma de reconstrucción de ligas en caso de pérdida de alguna de ellas, y así asegurar la integridad de la información estructural de la base de datos.

REGISTRO EN UN ARCHIVO DE DETALLE



TI ----> TIPO DE REGISTRO

ACA ----> APUNTADOR CADENA ANTERIOR

ACP ----> APUNTADOR CADENA POSTERIOR

NR ----> NUMERO DE REGISTRO

IV.3 ARCHIVOS MAESTROS.

Cuando los datos son puestos en un archivo maestro, el valor del campo llave es convertido a un número de registro relativo por una rutina de HASHING. Esta técnica será enfocada posteriormente en éste capítulo. El primer registro enviado a una localidad 'x' por el algoritmo se llamará registro primario y el segundo se llamará sinónimo, un sinónimo será puesto en el siguiente registro vacío, y es ligado con el primario asociado a través de una cadena de sinónimos. Las cadenas de sinónimos están formadas por un apuntador posterior y otro anterior, los registros conteniendo sinónimos son llamados registros secundarios.

El manejador localizará el primer o último registro de datos de una cadena utilizando una cabeza de cadena. La cabeza de cadena, para una cadena en particular, es almacenada con la información en el archivo maestro correspondiente, cuyo valor de campo llave es el mismo que el valor del campo llave del detallado que define la cadena. Cada cadena es de dos palabras y consiste de dos enteros;el número de registro del primer registro de la cadena (o cabeza) y el número de registro del último registro de la cadena (o pie).

La forma del medio de registro de un archivo maestro será mostrada en la siguiente figura.

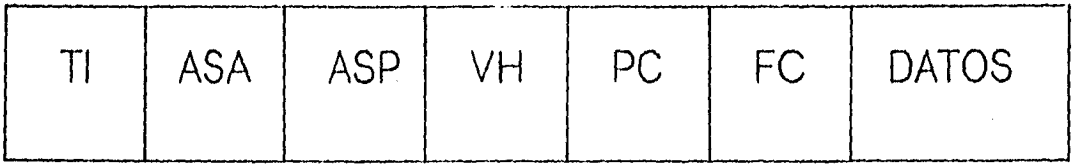
La primer palabra es el tipo de registro y será cero, si el registro está vacío, 1 si el registro es un registro primario, y -1 si el registro es un registro secundario(o sinónimo), las siguientes 2 palabras son los apuntadores anterior y posterior a la cadena de sinónimos, la siguiente palabra es el valor HASHING del registro con lo que sabremos si un sinónimo en la cadena es auténtico, es decir tiene el mismo valor hashing que la entrada primaria, o no, es decir no tiene el valor hashing de la cabeza pero forma parte de esta, pues su lugar estaba ocupado por un sinónimo a su vez. Las dos siguientes palabras son los apuntadores anterior y posterior respectivamente, a la cabeza y fin de una ca-

dena formada con un archivo de detalle. Estas dos últimas palabras valdrán 0 si no hay ocurrencias de la misma llave con el archivo de detalle asociado.

El tamaño de un registro en un archivo maestro será pues 5 + n, donde n es el número de palabras que ocuparán los datos.

Las cadenas de sinónimos son doblemente ligadas para facilitar el movimiento de datos cuando se da de baja una llave dentro de un archivo maestro; pues es necesario en ocasiones recorrer registros, o cuando se da de alta un sinónimo, pues se tiene que recorrer la cadena en ambos sentidos para asegurar si algún elemento en la cadena fue un sinónimo dado de baja y se puede ocupar su lugar.

REGISTRO EN UN ARCHIVO MAESTRO



TI ----> TIPO DE CAMPO
ASA ----> APUNTADOR SINONIMO ANTERIOR
ASP ----> APUNTADOR SINONIMO POSTERIOR
VH ----> VALOR HASHING
PC ----> PRINCIPIO DE CADENA
FC ----> FIN DE CADENA

IV.4 ARCHIVO RAIZ.

El archivo Raíz contendrá una descripción de la base de datos. Es almacenado en un archivo de disco cuyo nombre será el dado como nombre de la base de datos en el esquema. Cuando una base de datos es abierta, el archivo Raíz es copiado a memoria en una area de datos del programa de aplicación, cuando el archivo Raíz es leído parte de la información se vuelve dinámica y el archivo Raíz será referenciado como tabla de corrida de la base de datos.

El archivo Raíz consistirá de varias tablas y son:

+ El bloque de control de la base de datos BCBD, que contiene información general de la base de datos y apuntadores a otras tablas.

+ La Tabla de Campos Dato TCD contiene un registro por cada campo contenido en la base de datos.

+ La Tabla de Control de los Archivos de Datos TCAD contiene un registro para cada archivo de la base de datos con información general de este.

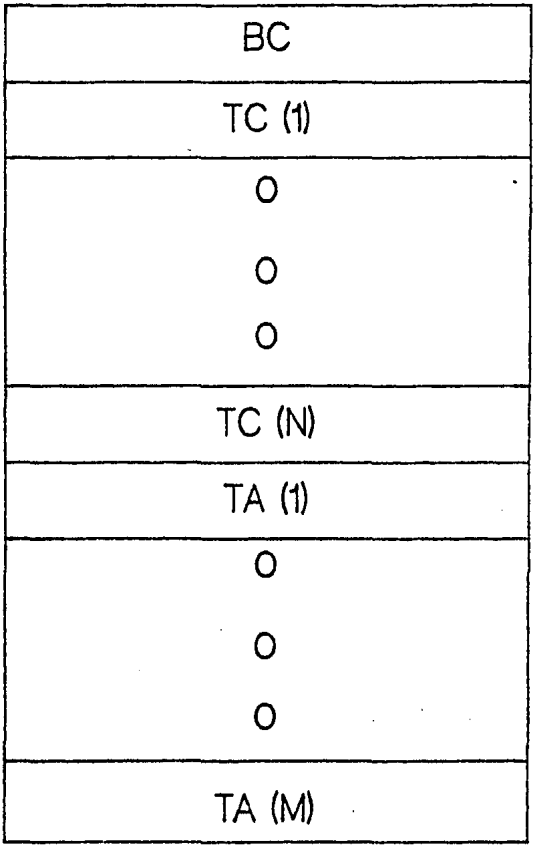
La estructura de éstas tablas será mostrada posteriormente.

IV.5 BLOQUE DE CONTROL DE LA BASE DE DATOS.

El bloque de control de la base de Datos tendrá información referente a la base de datos, e informacion del archivo Raíz del cual forma parte. Existe en el archivo Raíz solo un bloque de control, pues éste es único para cada base de datos.

Los asteriscos en las figuras de las tablas de campos dato, archivos de datos y bloque de control (en las siguientes hojas) denotan palabras reservadas para futuro crecimiento de la base, es decir si se desean agregar nuevas capacidades a esta.

FORMA GENERAL DE UN ARCHIVO RAIZ



BC --> BLOQUE DE CONTROL DE LA BASE DE DATOS
TC --> TABLA DE CAMPOS
TA --> TABLA DE ARCHIVOS

N ---> NUMERO DE CAMPOS EN LA BASE
M ---> NUMERO DE ARCHIVOS EN LA BASE

IV.6 TABLA DE CAMPOS DATO.

Esta base de datos contendrá 14 palabras relacionadas con cada campo de la base de datos. El formato de esta tabla se muestra en la siguiente figura. Los registros de ésta tabla serán ordenados por número de campo, donde el primero será el número 1. La cuenta de archivo de datos es el número de archivos en los cuales el campo aparece. El número de Archivo de datos es el número del primer archivo de datos donde aparece el campo. El número de archivo proporcionará un punto de inicio en la tabla de información de archivos de datos, para cuando se va a iniciar una búsqueda y la cuenta de archivo de datos dará la pauta de cuando terminar dicha búsqueda.

El tipo de campo podrá ser real, entero o alfanumérico. Los asteriscos en las figuras denotan palabras reservadas para futuro crecimiento de la base, es decir si se desean agregar nuevas capacidades a esta.

TABLA DE CAMPOS DATO

1	NOMBRE DEL CAMPO CAR. 1 Y 2
2	NOMBRE DEL CAMPO CAR 3 Y 4
3	NOMBRE DEL CAMPO CAR. 5 Y 6
4	TIPO DEL CAMPO
5	*
6	# DEL ARCHIVO AL QUE PERTENECE
7	# DE OCURRENCIAS
8	TAMANO DEL CAMPO
9	NUMERO DE CAMPO ASIGNADO
10	NUMERO DE LIGAS
11	ARCHIVO CON EL QUE LIGA
12	*
13	*
14	*

* CAMPO RESERVADO PARA FUTURO CRECIMIENTO

IV.7 TABLA DE ARCHIVOS DE DATOS.

Esta tabla da un registro de 14 palabras para cada archivo de datos en la base. La forma de ésta tabla es mostrada a continuación. De la palabra 1 a la 3 tenemos el nombre del archivo de datos, en la palabra 5 tenemos el tipo de archivo del que se trata, que puede ser Maestro automático, Maestro Manual o Detallado, la palabra 6 nos define el tamaño del registro del archivo, tomando en cuenta el medio del registro, la palabra 7 contendrá la capacidad en número de registros del archivo de datos, en la palabra 9 tendremos el número de archivo asignado en la generación de la base de datos.

Los asteriscos denotan palabras reservadas para futuro crecimiento de la base, es decir sí se desean agregar nuevas capacidades a esta.

>>> Una palabra = 2 bytes en la HP-1000 donde originalmente se creó el software, aunque ésto será ajustado para portabilidad con otras máquinas.

TABLA DE ARCHIVOS DE DATOS

1	NOMBRE DEL ARCHIVO CAR. 1 Y 2
2	NOMBRE DEL ARCHIVO CAR. 3 Y 4
3	NOMBRE DEL ARCHIVO CAR. 5 Y 6
4	UNIDAD DONDE RESIDE
5	TIPO DE ARCHIVO
6	*
7	CAPACIDAD DEL ARCHIVO
8	NUMERO DEL CAMPO LLAVE
9	# DE ARCHIVO ASIGNADO
10	ULTIMO REGISTRO ACCESADO
11	SIG. REGISTRO EN CADENA
12	NUMERO DE REGISTROS LIBRES
13	SIGUIENTE REGISTRO LIBRE
14	*

* CAMPO RESERVADO PARA FUTURO CRECIMIENTO

IV.8 ARCHIVO RAIZ

Hasta éste punto hemos apreciado las partes que conforman al archivo Raíz (Bloque de control, Tabla de Campos, Tabla de Archivos), por lo que a continuación veremos como se forma el archivo Raíz, el cual le da el nombre a la Base de Datos.

El archivo Raíz tendrá un Bloque de Control pues este bloque debe ser único para cada Base de datos, también contará con tantas tablas de archivos, como archivos se deseen tener en la Base de datos y con tantas tablas de Campos como campos habrá distribuidos en los archivos de la base de datos.

El tamaño de un archivo Raíz será entonces : 14 + 14*n + 14*m .

Donde n es el número de Campos y m es el número de archivos que tendrá la base de datos.

El archivo Raíz tendrá 2 estados, uno en disco y otro en memoria, cuando la base de datos no sea accesada estará en el disco, y cuando sea accesada se subirá a memoria en los Buffers de las rutinas de la Base de datos. Esto es con la finalidad de minimizar los accesos a disco y aumentar la velocidad de manejo de nuestras bases de datos.

BLOQUE DE CONTROL DEL ARCHIVO RAIZ

1	NOMBRE DE LA BASE CAR. 1 Y 2
2	NOMBRE DE LA BASE CAR. 3 Y 4
3	NOMBRE DE LA BASE CAR. 5 Y 6
4	CODIGO DE SEGURIDAD
5	UNIDAD DE DISCO
6	NUMERO DE CAMPOS
7	APUNTADOR A TABLA DE CAMPOS
8	NUMERO DE ARCHIVOS
9	APUNTADOR A TABLA DE ARCHIV.
10	INDICADOR DE BASE ABIERTA
11	ULTIMO REGISTRO LEIDO
12	*
13	*
14	*

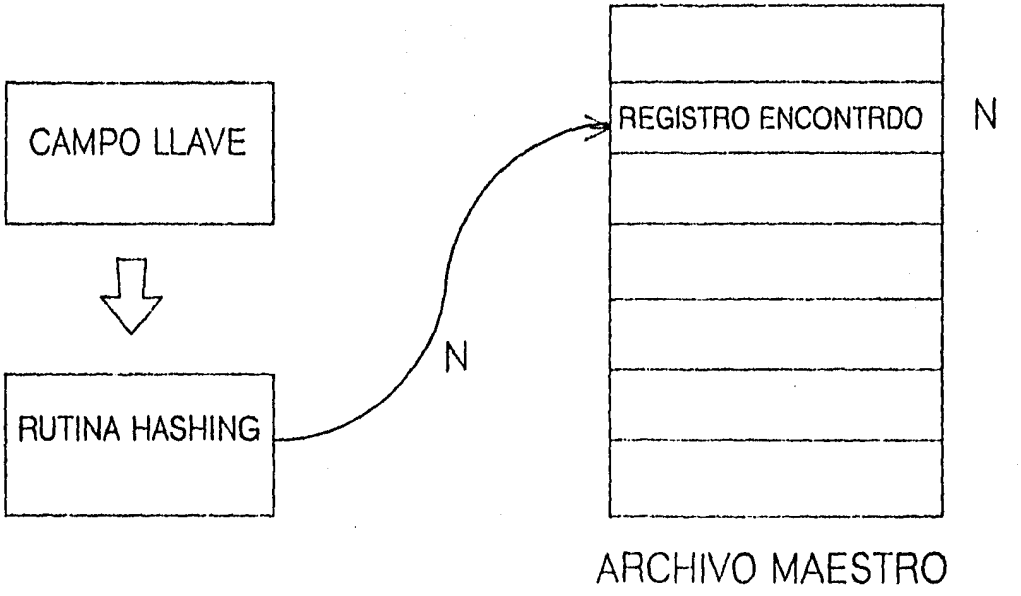
* CAMPO RESERVADO PARA FUTURO CRECIMIENTO

IV.2 TECNICAS PARA LECTURA POR LLAVE

El manejador de la Base de Datos utilizará una técnica de dispersión para calcular las localidades donde residirán los registros de datos en un archivo maestro. Consistirá en tomar el campo llave y en base a su longitud y recorrerlo tomando los caracteres que lo forman.

Al ir tomando los caracteres se obtendrá su valor numérico en la tabla de caracteres Ascii y se irá acumulando en una sumatoria. Al final del recorrido de la palabra tendremos un número entero, el cual podría ser mayor al número de registros del archivo Maestro en cuestión, por lo que se hará el módulo del resultado obtenido con la capacidad del archivo lo que nos distribuirá los registros dentro del archivo Maestro.

FUNCIONAMIENTO GENERICO DE UNA RUTINA HASHING



N --> NUMERO DE REGISTRO

IV.10 COLISIONES

En algunos casos dos llaves diferentes nos podrán dar como resultado la misma dirección dentro de un mismo archivo maestro, por ejemplo la llave ABC y la llave BCA, por lo que se implementó una técnica para manejar estos casos, a los que llamaremos secundarios migratorios o sinónimos, un sinónimo tendrá un valor -1 en su marca, lo que facilitará al manejador detectarlo. Existe la posibilidad de que un sinónimo en una cadena de sinónimos no tenga el valor hashing que la entrada primaria, pues tal vez fue resultado de haber encontrado ocupada su localidad por otro sinónimo, y el criterio del manejador es colocarlo a continuación de la cadena que ocupaba su lugar.

En éstas circunstancias, las rutinas para dar de baja tendrán que observar ésta posibilidad, para que si se da de baja un registro no se rompa la cadena de sinónimos, imposibilitando la búsqueda de los sinónimos no auténticos. para éstos fines es útil el valor hashing dentro del medio de registro de un archivo maestro, si vale -99 quiere decir que un sinónimo fue dado de baja en ese lugar y que podrá ser ocupado solo por sinónimos que tengan el mismo valor hashing de la entrada primaria.

IV.11 EJEMPLO DE LA RUTINA DE HASHING

A continuación mostraremos un ejemplo de la rutina de Hashing utilizada para éste manejador de Bases de Datos.

La fórmula utilizada será la mostrada en la siguiente figura.

RUTINA DE HASHING

```
FOR I := 1 TO N DO
    D := ORD (LLAVE(I)) + D
(ENDDO)
D := (D MOD CAP) + 1
```

N --> NUMERO DE BYTES EN LA LLAVE

D --> DIRECCION DEL REGISTRO

I --> CONTADOR

CAP > CAPACIDAD DEL ARCHIVO EN REGISTROS

CON 'ORD' SE OBTIENE EL VALOR ASCIIDE UN CARACTER

CON 'MOD' SE OBTIENE UN REGISTRO DENTRO DEL ARCHIVO

Donde : D será la dirección donde quedará el registro.
N es el número de caracteres de la llave.
CAP es la capacidad del archivo en número de registros.

Supongamos que tenemos una Llave de 4 caracteres alfanu méricos (ABCD), y el archivo maestro tiene una capacidad de 100 registros.

+ El valor ASCII de una A es 65
+ El valor ASCII de una B es 66
+ El valor ASCII de una C es 67
+ El valor ASCII de una D es 68

+ D será 67, es decir el dato irá al registro 67 del archivo.

TEMA V RUTINAS PRINCIPALES DEL MANEJADOR DE BASES DE DATOS

Después de que la base de datos ha sido diseñada y el archivo Raíz con sus archivos de datos han sido creados, se procederá a escribir programas de aplicación los cuales serán utilizados para leer y escribir información en la base de datos. Estos programas serán escritos en un lenguaje de alto nivel, como FORTRAN o PASCAL, los cuales utilizarán llamadas de las librerías de la Base de datos, habra 10 rutinas para éstos fines.

ABRBD Inicia el acceso a la base de datos y verifica si la base está disponible.

LEEBD Lee registros en archivos maestros.

LEEBDD Lee registros en archivos de detalle.

ACTBD Modifica los valores de los campos datos en un registro determinado. (Archivos Maestros)

ACTBDD Modifica los valores de los campos datos en un registro determinado. (Archivos de Detalle)

PONBD Agrega nuevos registros a la base de datos, en archivos maestros.

PONBDD Agrega nuevos registros a la base de datos, en archivos de detalle.

BORBD Borra registros de la Base de Datos en Archivos Maestros.

BORBDD Borra registros de la Base de Datos en Archivos de Detalle.

CIEBD Cierra los Archivos de la Base de Datos.

REPER Reporta el error cometido de acuerdo a un código de error.

La información específica sobre los parámetros de cada llamada será discutida posteriormente, junto con una

descripción general de la misma.

V.1 APERTURA DE LA BASE DE DATOS.

Antes de que un usuario de la Base pueda accesarla, deberá abrir la con la llamada ABRBD. En la acción de abrir la base la llamada ABRBD establece una liga entre la Base de Datos y el programa de aplicación de la siguiente manera.

+ Verificando el derecho de uso de la base bajo condiciones de seguridad provistas por los archivos y el sistema.

+ Cargando la tabla de corrida a memoria para una ejecución más rápida.

+ Abriendo el archivo Raíz y construyendo una tabla de corrida que será utilizada por las rutinas de la base para ser ejecutadas. La tabla de corrida contiene información estática y dinámica sobre la Base de Datos. La información inicial de dicha tabla es obtenida inicialmente del archivo Raíz y modificada con el uso del sistema.

V.2 TABLA DE CORRIDA.

La tabla de corrida es una tabla de información relativa a la base de datos, utilizada por las rutinas del manejador para controlar acceso a la base. La tabla de corrida es creada cuando la base es abierta y borrada cuando la base es cerrada. Durante su existencia se encuentra en los buffers de las rutinas del manejador, que están a su vez en la partición del programa de aplicación.

V.3 LECTURA DE DATOS.

Cuando se lea la Base, el usuario determinará en que archivo y que registro de dicho archivo deberá ser accesado. Los campos de dicho registro podrán ser entonces leídos. La forma en que el programa de aplicación los manejará es definida por el usuario en el mismo.

para entender la manera en que los registros serán accesados, cabe recordar la estructura de la información en la Base. Cada archivo de la Base consiste en un Archivo de disco, el cual está constituido por registros, donde se encuentra la informacion de los campos a su vez. El primer registro en el archivo es el registro "1" y el enésimo registro es el registro "n", donde n es la capacidad de dicho archivo en registros.

En un momento determinado un registro puede o no contener información. El sistema mantiene información de cual registro contiene datos y cual no.

V.3.1 LIGA ACTUAL

El sistema mantiene información sobre la liga que se está trabajando. Esta liga es primero establecida por la rutina LEEBD y cada vez que se éste leyendo en forma encadenada, el manejador almacenará los apuntadores al registro anterior y al registro posterior.

V.3.2 MODOS DE LECTURA.

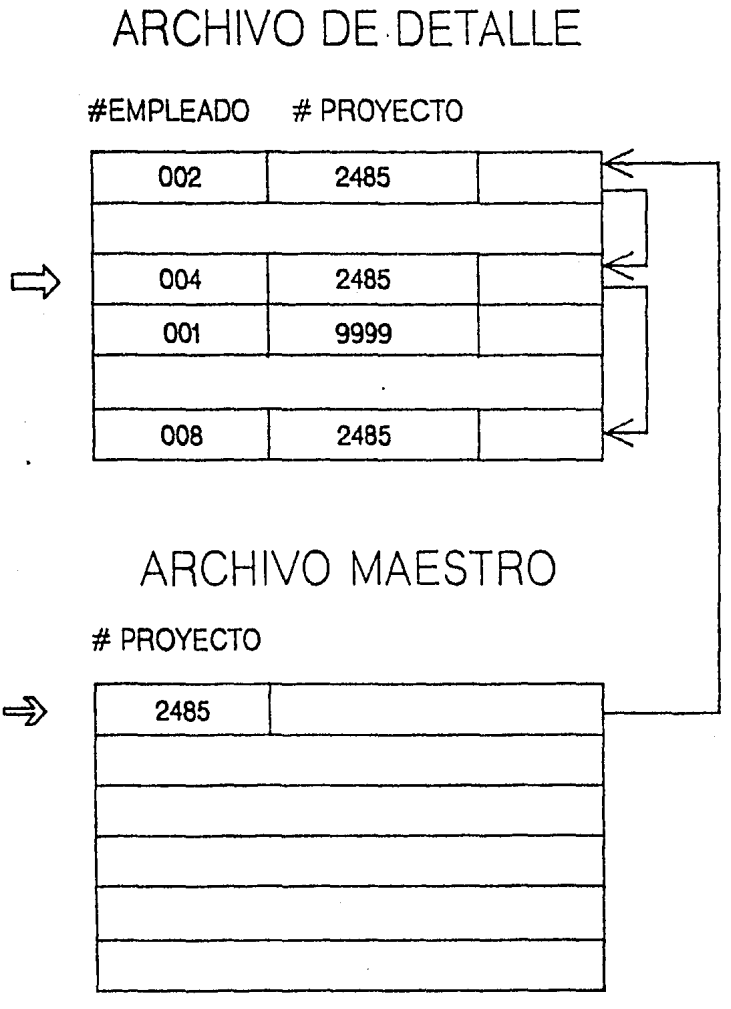
Los métodos para accesar registros pueden ser los siguientes.

+ Lectura encadenada.

+ Lectura directa

+ Lectura por llave

Todos éstos métodos son permitidos a través del manejador de la base con las rutina LEEBD y LEEBDD.


ARCHIVO DE DETALLE
#EMPLEADO
PROYECTO
002
2485
004
2485
001
9999
008
2485
ARCHIVO MAESTRO
PROYECTO
2485
LECTURA ENCADENADA
LECTURA POR LLAVE
LECTURA DIRECTA

V.3.3 LECTURA ENCADENADA

Este modo de lectura es utilizado para traer información sobre el siguiente registro en relación al que se acaba de leer.

Las lecturas encadenadas solo ocurrirán en archivos de detalle y requiere que el usuario utilice el llamado LEEBDD con anterioridad para localizar el primer registro y luego leerlo con la misma llamada. El programa de aplicación determina la llave del archivo de detalle que será accesada y que definirá la cadena. El manejador determina que archivo maestro está ligado con la llave mencionada y localiza el registro en dicho maestro, cuyo valor de llave coincida con el especificado. La información localizada contiene apuntadores al primero y al último registro de la cadena, ésta información se conserva y nos define la liga actual.

Una vez que la liga y la cadena se han localizado para un archivo de datos, el programa de aplicación puede hacer lecturas encadenadas, utilizando LEEBDD , para accesar los datos. Se permiten lecturas encadenadas ascendentes. Al final de la cadena se recibirá un mensaje indicando el fin de esta, en uno de los parámetros del llamado.

Las lecturas encadenadas son particularmente útiles cuando el usuario quiere traer información sobre eventos relacionados entre si.

V.3.4 LECTURA DIRECTA.

Otro método para seleccionar ocurrencias de un dato a ser leído, será especificando el número de registro. Este método se llama lectura directa. Si un registro existe en la dirección especificada por un programa de aplicacion, el manejador regresará los valores de los campos de dicho registro especificados en el buffer de la llamada. De lo contrario el programa será notificado con un código de error.

Este método de acceso puede ser usado con archivos de

tipo detalle dentro de la base y puede ser útil cuando el programa de aplicación tiene las direcciones de los registros que se desean accesar.

Por ejemplo si un programa ha estado leyendo secuencialmente un archivo y ha seleccionado ciertos registros, puede hacer una lectura directa con el apuntador a dichos registros almacenado previamente, para un reporte específico.

V.3.5 LECTURA POR LLAVE.

Este método permite el acceso de un registro del archivo maestro con el valor de un campo llave en particular, ésta lectura solo puede ser usada con archivos maestros. Puede ser muy útil para traer información de un registro específico en un archivo. Por ejemplo si se deseara traer información de un empleado con clave XXXX del archivo maestro de personal;

V.4 ACTUALIZACION DE DATOS.

El manejador permitirá modificar todos los valores de los campos, siempre y cuando éstos no sean campos llave. Antes de utilizar la rutina ACTBDD para actualizar un archivo de detalle, el registro adecuado debe ser encontrado con la rutina LEEBDD, en el caso de la rutina ACTBD el registro se encuentra con el valor de la llave.

Cuando un programa invoca las llamadas ACTBD y ACTBDD, se le especificará el nombre del archivo de datos, y el buffer conteniendo los nuevos valores de los campos.

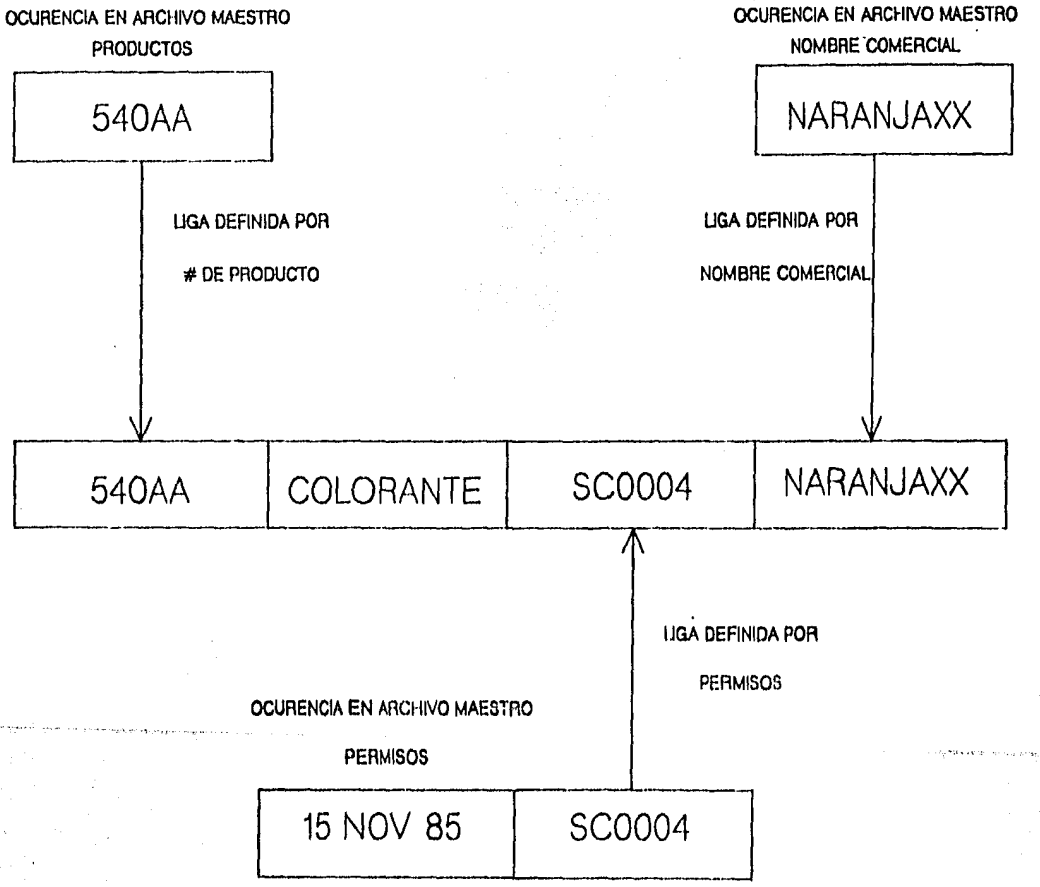
V.5 ALTA DE DATOS EN LA BASE.

Los datos se dan de alta en la base de datos uno por uno, utilizando las llabadas PONBD y PONBDD. Los datos podrán ser dados de alta en cualquier tipo de archivo dependiendo del tamaño, salvo en los maestros automáticos, donde se dan de alta solo con el hecho de empezar una nueva liga con el archivo de detalle.

para agregar un registro se deberá especificar el nombre del archivo de datos, una lista de los campos del archivo, y un buffer con el valor de dichos campos. Será forzoso por lo menos dar el valor del campo llave para poder agregar algún registro.

V.5.1 SECUENCIA PARA AGREGAR DATOS.

Antes de que se pueda agregar un registro a un archivo detallado ligado a algún maestro manual, el archivo maestro deberá contener, de antemano, algun registro con un campo llave del mismo valor que el que se pretende dar de alta en el de detalle. Si hay más de un archivo maestro manual ligado a un detallado, cada uno de ellos deberá contener un campo llave de valor idéntico al registro del detallado correspondiente.


OCURENCIA EN ARCHIVO MAESTRO
PRODUCTOS
540AA
OCURENCIA EN ARCHIVO MAESTRO
NOMBRE COMERCIAL
NARANJAXX
LIGA DEFINIDA POR
DE PRODUCTO
LIGA DEFINIDA POR
NOMBRE COMERCIAL
540AA
COLORANTE
SC0004
NARANJAXX
LIGA DEFINIDA POR
PERMISOS
OCURENCIA EN ARCHIVO MAESTRO
PERMISOS
15 NOV 85
SC0004

V.6.2 CAMPOS LLAVE.

El manejador checará el campo llave antes de agregar un registro a la base de datos. Sí el archivo es un archivo maestro manual, el manejador revisará que ese valor sea único en el archivo de datos. Sí el archivo de datos es Detallado el manejador checará que exista una llave con el mismo valor a la que se pretende dar de alta dentro del archivo maestro correspondiente. También revisa que haya espacio en el archivo automático, sí la llave correspondiente en el detallado no existe.

V.6 BORRADO DE REGISTROS DE LA BASE DE DATOS.

para borrar un registro de un archivo de datos, el registro a borrar debe ser primeramente localizado. Se permiten dos métodos para ello.

1) Borrar el registro con la rutina BORBD.

2) Borrar el registro con una combinación de LEEBDD y BORBDD en el caso de archivos de detalle.

para Borrar con las rutinas BORBD y BORBDD se especificará el nombre del archivo de datos. El manejador revisará sí la Base de datos fue abierta previamente para dar de baja los registros.

Sí el registro que se borra es el último de una liga relacionada con un archivo automático y el resto de las ligas tampoco tienen ocurrencias en otros archivos de detalle, el manejador se encargará de borrar el registro del archivo automático.

Sí el registro que se desea borrar está en un archivo maestro manual, el manejador revisará que sus ligas con algun archivo de detalle estén vacías, de lo contrario, no permitirá dar de baja tal registro.

V.7 CERRADO DE LA BASE DE DATOS.

Después de haber realizado todas las operaciones deseadas dentro de la base de datos, se utilizará la ru

tina CIEBD para concluir el acceso a ella, cuando esta rutina es utilizada todos los archivos de la base son cerrados.

Será importante utilizar ésta rutina cada vez que se termine un proceso de acceso a la base, para asegurar el cerrado ordenado de los diferentes archivos de la base.

V.8 REVISION DE CODIGOS DE ESTADO.

Cada vez que una subrutina de la base es llamada, el manejador regresa información de estado en una variable especifica en el programa de aplicación. La cual contiene un valor que especifica el estado de la operación, será cero si la operación fue exitosa y diferente de cero si hubo algún error en ella. Este código de estado deberá ser revisado después de cada operación. Un error que no sea detectado en el momento de su ocurrencia podrá acarrear severos errores en el futuro. Conjuntamente con los errores el manejador enviará cierta información de utilidad para el control de los programás de aplicación.

Los códigos de estado podrán ser interpretados con la rutina del manejador REPER, la cual desplegará un mensaje describiendo el error cometido y recibe como parámetro el código de error.

V.9 PARAMETROS DE LLAMADO.

En la utilización de rutinas del manejador se deberá enviar, como parámetro, cierta información de acuerdo a la llamada que se invoque.

Genéricamente los primeros tres parámetros en las llamadas del manejador serán, Nombre de la Base de Datos, código de seguridad de los archivos de la base de datos y la unidad de disco donde éstos residen. El cuarto parámetro será el nombre del archivo al cual se está accesando en particular, salvo en las rutinas CIEBD y ABRBD donde el cuarto parámetro será la variable de estado, utilizada para interpretación de errores. El quin

to parámetro será el nombre de la llave del archivo que se está accesando. El sexto parámetro, en las llamadas para archivos de detalle, es el número de registro que se va a accesar, salvo en la llamada LEEBDD cuyo valor no debe ser utilizado por el usuario. El septimo parámetro en la llamada LEEBDD es el número de registro que se acaba de leer.

En el caso de la rutina REPER el único parámetro será el código de error recibido de alguna de las demás rutinas.

A continuación se definirán algunos de los parámetros que nos serán utiles en las llamadas al manejador.

V.9.1 BASE.

Será un arreglo conteniendo el nombre que describe a la base de datos que deseamos accesar, el cual deberá ser de 6 caracteres máximo. Con ésto localizaremos el archivo Raíz de la base en el disco.

```
DIMENSION Nombas(3)
DATA Nombas/6HINVEN /
```

El ejemplo anterior en FORTRAN nos denota como en un arreglo de 6 caracteres se asigna el nombre de una base de Inventarios (INVEN).

El valor de la variable Base será definido en la apertura y no deberá se cambiado mientras se esté accesando la misma base, pues será utilizada en el resto de los llamados.

V.9.2 CODIGO

Esta variable será utilizada para pasar al manejador de la base el código de seguridad de sus archivos. Esto con la finalidad de que no cualquier usuario pueda abrir la Base aunque sepa el nombre.

El código deberá ser un arreglo de dos caracteres alfanuméricos.

V.9.3 CARTUCHO.

Este parámetro nos dirá en que porción de disco residen los archivos de la Base de Datos, por lo que se podrán tener Bases con el mismo nombre y por residir en lugares diferentes, podrán tener información diferente.

Este parámetro será un arreglo de dos dígitos.

V.9.4 ESTADO.

Es una variable mediante la cual el manejador nos regresará información suficiente para detectar si hubo error en la aplicación de una de las rutinas del manejador.

El valor obtenido será un número entero, el cual, despues podrá ser utilizado para desplegar una explicacion del error, con ayuda de la rutina REPER.

V.9.5 BUFFER.

La variable Buffer será un acumulador o arreglo utilizado para pasar información del usuario hacia la Base y de la base hacia el usuario. El tamaño de este acumulador variará de acuerdo a la información que se desee transmitir. El uso de éste acumulador dependerá del tipo de rutina que se desee utilizar y será descrito posteriormente, en la definición de los parámetros de cada llamado al manejador.

V.9.6 CAMPO.

La variable que tome la posición de campo es un arreglo de seis caracteres, que contendrá como información, el nombre del campo llave sobre el cual se realizará una búsqueda.

V.9.7 ARGUMENTO.

Es una variable donde se almacenará el número de registro relativo correspondiente a un registro de la base que se desee accesar directamente.

V.10 PROCEDIMIENTO PARA EL USO DE RUTINAS DEL MANEJADOR.

A continuación se discutirá el uso de los diferentes parámetros en las distintas utilerias del manejador. Todos los parámetros mencionados deberán aparecer en cada llamado, pues su significado es determinado por su posición.

V.10.1 PARAMETROS NO UTILIZADOS.

Cuando se invoque alguna rutina del manejador, alguno de los parámetros podrá ser ignorado. Sin embargo deberá ser listado en el llamado.

V.10.2 APERTURA DE LA BASE DE DATOS.

ABRBD(BASE, CODIGO, CARTUCHO, ESTADO)

Inicia el acceso a la base de datos y revisa sí la Base no está abierta.

V.10.2.1 PARAMETROS.

BASE>> Es un arreglo que contendrá una cadena ASCII de caracteres con el nombre de la base de datos que se desee abrir, sí la base se abre exitosamente, ABRBD escribe el número de la base de datos correspondiente en el arreglo, por lo que no debe ser alterado su valor mientras se deseen accesar datos de la base.

Código>> Es un arreglo conteniendo el código de seguridad de la base de datos y no deberá ser modificado en el resto de los llamados a la base de datos.

CARTUCHO>> Es un arreglo conteniendo la porción de disco donde reside la base de datos y no deberá ser modificado en el resto de los llamados a la base.

ESTADO>> Es una variable entera donde recibiremos un código de error el cual nos definirá sí la apertura de la base fue exitosa o no.

V.10.3 ACCESO DE REGISTROS DE ARCHIVOS MAESTROS.

LEEBD(BASE, Código, CARTUCHO, ARCHIVO, CAMPO, BUFFER, ESTADO)

Esta rutina se utilizará para leer datos de un archivo maestro, utilizando un método apoyado en el algoritmo de dispersión .

V.10.3.1 PARAMETROS.

BASE>> Es un arreglo, utilizado como nombre de la base de datos desde la apertura, el cual no debe haber sido alterado, como se indicó, por el usuario.

CODIGO>> Es un arreglo conteniendo el código de seguridad de la base de datos, el cual no debe haber sido alterado a partir de la apertura de la base de datos.

CARTUCHO>> Es un arreglo conteniendo la porción de disco donde reside la base de datos, el cual no debe haber sido alterado a partir de la apertura de la base de datos.

ARCHIVO>> Es una variable conteniendo el nombre del archivo de la base de datos, del cual deseamos leer y deberá ser una cadena de 6 caracteres alfanuméricos.

CAMPO>> Es una arreglo conteniendo el nombre del campo llave del archivo que estamos accesando y deberá ser una cadena de 6 caracteres alfanuméricos.

BUFFER>> Es un arreglo en el cual LEEBD regresará el contenido con catenado de los campos residentes en el registro leído, en el orden que fueron definidos en la generación de la base.

ESTADO>> Es una variable donde LEEBD regresa un número indicando si se cometió algún error (Estado <> 0) o si el llamado fue Exitoso (Estado = 0).

V.10.4 ACCESO DE REGISTROS DE ARCHIVOS DETALLADOS.

LEEBDD<BASE, CODIGO, CARTUCHO,ARCHIVO,CAMPO, BUFFER, POS,ARGUMENTO>

Esta rutina se utilizará para leer datos de un archivo de detalle y nos dará la posibilidad de hacer lecturas encadenadas.

V.10.4.1 PARAMETROS.

BASE>> Es un arreglo, utilizado como nombre de la base de datos desde la apertura, el cual no debe haber sido alterado, como se indicó, por el usuario.

CODIGO>> Es un arreglo conteniendo el código de seguridad de la base de datos, el cual no debe haber sido alterado a partir de la apertura de la base de datos.

CARTUCHO>> Es un arreglo conteniendo la porción de disco donde reside la base de datos, el cual no debe haber sido alterado a partir de la apertura de la base de datos.

ARCHIVO>> Es una variable conteniendo el nombre del archivo de la base de datos, del cual deseamos leer y deberá ser una cadena de 6 caracteres alfanuméricos.

CAMPO>> Es una arreglo conteniendo el nombre del campo llave del archivo que estamos accesando y deberá ser una cadena de 6 caracteres alfanuméricos.

BUFFER>> Es un arreglo en el cual LEEBDD regresará el contenido con catenado de los campos residentes en el registro leído, en el orden que fueron definidos en la generación de la base.

ESTADO>> Es una variable donde LEEBDD regresa un número indicando sí se cometió algún error (Estado <> 0) o sí el llamado fue Exitoso (Estado = 0).

POS>> Es una variable enterá reservada para el manejador, es decir, que el usuario no deberá alterarla.

ARGUMENTO>> Es una variable enterá donde enviaremos y recibiremos información.

1 > Al recibir tendremos en el el número de registro recien accesado.

2 > Al enviar deberá cumplir las siguientes condiciones de acuerdo a lo que deseemos realizar.

a) Lectura encadenada :

Argumento = 0 para encontrar la cabeza de la cadena sí a partir de éste momento no alteramos el valor de Argumento y seguimos leyendo nos traerá el siguiente registro en la cadena.

Argumento <> 0 para leer el siguiente registro en la cadena.

V.10.5 ACTUALIZACION DE REGISTROS EN ARCHIVOS MAESTROS.

ACTBD(BASE, CODIGO, CARTUCHO, ARCHIVO, CAMPO, BUFFER, ESTADO)

Esta rutina modificará los valores de los campos del registro correspondiente a una llave dada en un archivo Maestro.

V.10.5.1 PARAMETROS.

BASE>> Es un arreglo, utilizado como nombre de la base de datos desde la apertura, el cual no debe haber sido alterado, como se indicó, por el usuario.

CODIGO>> Es un arreglo conteniendo el código de seguridad de la base de datos, el cual no debe haber sido alterado a partir de la apertura de la base de datos.

CARTUCHO>> Es un arreglo conteniendo la porción de disco donde reside la base de datos, el cual no debe haber sido alterado a partir de la apertura de la base de datos.

ARCHIVO>> Es una variable conteniendo el nombre del archivo de la base de datos, del cual deseamos leer y deberá ser una cadena de 6 caracteres alfanuméricos.

CAMPO>> Es una arreglo conteniendo el nombre del campo llave del archivo que estamos accesando y deberá ser una cadena de 6 caracteres alfanuméricos.

BUFFER>> Es un arreglo en el cual se enviará a la Base de Datos la modificación deseada, los campos van concatenados en el orden que fueron definidos en la generación de la Base de Da tos.

ESTADO>> Es una variable donde ACTBD regresa un número indicando si se cometió algún error (Estado <> 0), o si el llamado fue Exitoso (Estado = 0).

V.10.6 ACTUALIZACION DE REGISTROS EN ARCHIVOS DETALLADOS

ACTBDD(BASE,Código,CARTUCHO,ARCHIVO,CAMPO,BUFFER,ESTADO,ARGUMENTO)

Esta rutina modificará los valores de los campos del registro correspondiente a un registro dado en un archivo Detallado.

V.10.6.1 PARAMETROS.

BASE>> Es un arreglo, utilizado como nombre de la base de datos desde la apertura, el cual no debe haber sido alterado, como se indicó, por el usuario.

Código>> Es un arreglo conteniendo el código de seguridad de la base de datos, el cual no debe haber sido alterado a partir de la apertura de la base de datos.

CARTUCHO>> Es un arreglo conteniendo la porción de disco donde reside la base de datos, el cual no debe haber sido alterado a partir de la apertura de la base de datos.

ARCHIVO>> Es una variable conteniendo el nombre del archivo de la base de datos, del cual deseamos leer y deberá ser una cadena de 6 caracteres alfanuméricos.

CAMPO>> Es una arreglo conteniendo el nombre del campo llave del archivo que estamos accesando y deberá ser una cadena de 6 caracteres alfanuméricos.

BUFFER>> Es un arreglo en el cual se enviará a la Base de Datos la modificación deseada, los campos van concatenados en el orden que fueron definidos en la generación de la Base de Datos.

ESTADO>> Es una variable donde ACTBDD regresa un numero indicando sí se cometió algun error (Estado <> 0), o sí el llamado fue Exitoso (Estado = 0).

ARGUMENTO>> Es una variable donde se envía un número entero indicando el número de registro del archivo de detalle que se va a modificar.

Por lo que antes se deberá haber leído el registro con un llamado LEEBDD, y tomado el valor regresado en su ARGUMENTO

V.10.7 ALTA DE REGISTROS EN ARCHIVOS MAESTROS.

PONBD(BASE, CODIGO, CARTUCHO, ARCHIVO, CAMPO, BUFFER, ESTADO)

Esta rutina agregará nuevos valores de campos a un archivo Maestro especificado. Utilizando el algoritmo de HASHING para encontrar espacios donde depositar los nuevos registros.

V.10.7.1 PARAMETROS.

BASE>> Es un arreglo, utilizado como nombre de la base de datos desde la apertura, el cual no debe haber sido alterado, como se indicó, por el usuario.

CODIGO>> Es un arreglo conteniendo el código de seguridad de la base de datos, el cual no debe haber sido alterado a partir de la apertura de la base de datos.

CARTUCHO>> Es un arreglo conteniendo la porción de disco donde reside la base de datos, el cual no debe haber sido alterado a partir de la apertura de la base de datos.

ARCHIVO>> Es una variable conteniendo el nombre del archivo de la base de datos, del cual deseamos leer y deberá ser una cadena de 6 caracteres alfanuméricos.

CAMPO>> Es una arreglo conteniendo el nombre del campo llave del archivo que estamos accesando y deberá ser una cadena de 6 caracteres alfanuméricos.

BUFFER>> Es un arreglo en el cual se enviará a la Base de Datos la información que se desea dar de alta.

Los campos van concatenados en el orden que fueron definidos en la generación de la Base de Datos.

ESTADO>> Es una variable donde PONBD regresa un número indicando sí se cometió algún error (Estado <> 0), o sí el llamado fue Exitoso (Estado = 0).

V.10.7.2 ARCHIVOS MAESTROS.

Cuando se agregue información a un archivo maestro las siguientes reglas aplicarán.

+ El Archivo deberá ser Maestro Manual.

+ El campo llave deberá ser referenciado y su valor en el BUFFER deberá ser único con respecto al resto de las llaves en el archivo.

+ Deberá haber espacio en el Archivo maestro para agregar un nuevo registro.

+ Los campos no referenciados serán llenados con ceros binarios.

V.10.8 ALTA DE REGISTROS EN ARCHIVOS DE DETALLE.

PONBDD(BASE,CODIGO,CARTUCHO,ARCHIVO,CAMPO,BUFFER,ESTADO,ARGUMENTO)

Esta rutina agregará nuevos valores de campos a un archivo Detallado especificado. Manejando apuntadores para encontrar espacios donde depositar los nuevos registros.

V.10.8.1 PARAMETROS.

BASE>> Es un arreglo, utilizado como nombre de la base de datos desde la apertura, el cual no debe haber sido alterado, como se indicó, por el usuario.

CODIGO>> Es un arreglo conteniendo el código de seguridad de la base de datos, el cual no debe haber sido alterado a partir de la apertura de la base de datos.

CARTUCHO>> Es un arreglo conteniendo la porción de disco donde reside la base de datos, el cual no debe haber sido alterado a partir de la apertura de la base de datos.

ARCHIVO>> Es una variable conteniendo el nombre del archivo de la base de datos, del cual deseamos leer y deberá ser una cadena de 6 caracteres alfanuméricos.

CAMPO>> Es una arreglo conteniendo el nombre del campo llave del archivo que estamos accesando y deberá ser una cadena de 6 caracteres alfanuméricos.

BUFFER>> Es un arreglo en el cual se enviará a la Base de Datos la información que se desea dar de alta.

Los campos van concatenados en el orden que fueron definidos en la generación de la Base de Datos.

ESTADO>> Es una variable donde PONBDD regresa un número indicando sí se cometió algún error (Estado <> 0), o sí el llamado fue Exitoso (Estado = 0).

V.10.8.2 ARCHIVOS DETALLADOS.

Cuando se agregue información a un archivo detallado las siguientes reglas aplicarán.

+ El Archivo deberá tener espacio libre para el nuevo registro.

+ Todos los campos llaves definidos para el registro deberán ser referenciados.

+ Cada archivo Maestro relacionado con el de detalle, deberá contener un campo llave de igual contenido que su campo llave de liga en el detallado.

+ Los campos no referenciados serán llenados con ceros binarios.

V.10.9 BAJA DE REGISTROS EN LA BASE DE DATOS.

BORBD(BASE, CODIGO, CARTUCHO, ARCHIVO, CAMPO, BUFFER, ESTADO)

Esta rutina borrará los valores de campos en archivo de Maestro especificado, utilizando el algoritmo de HASHING para encontrar el registro que se desee borrar.

V.10.9.1 PARAMETROS.

BASE>> Es un arreglo, utilizado como nombre de la base de datos desde la apertura, el cual no debe haber sido alterado, como se indicó, por el usuario.

CODIGO>> Es un arreglo conteniendo el código de seguridad de la base de datos, el cual no debe haber sido alterado a partir de la apertura de la base de datos.

CARTUCHO>> Es un arreglo conteniendo la porción de disco donde reside la base de datos, el cual no debe haber sido alterado a partir de la apertura de la base de datos.

ARCHIVO>> Es una variable conteniendo el nombre del archivo de la base de datos, del cual deseamos leer y deberá ser una cadena de 6 caracteres alfanuméricos.

CAMPO>> Es una arreglo conteniendo el nombre del campo llave del archivo que estamos accesando y deberá ser una cadena de 6 caracteres alfanuméricos.

BUFFER>> Es un arreglo en el cual se enviará a la Base de Datos la información que se desea boorrar. Los campos van concatenados en el orden que fueron definidos en la generación de la Base de Datos.

ESTADO>> Es una variable donde BORBD regresa un número indicando si se cometió algún error (Estado <> 0), o si el llamado fue Exitoso (Estado = 0).

V.10.9.2 ARCHIVOS MAESTROS.

Cuando se borre información de un archivo maestro la siguiente regla aplicará: toda la información de apuntadores para cadenas asociadas con el registro deberán indicar que la cadena está vacía. En otras palabras no deberá haber ocurrencias de la llave que se va a dar de baja en ningún archivo detallado asociado.

V.10.10 BORRADO DE REGISTROS EN ARCHIVOS DETALLADOS

BORBDD(BASE,CODIGO,CARTUCHO,ARCHIVO,CAMPO,BUFFER,ESTADO,ARGUMENTO)

Esta rutina Borrará los registros especificados con el parámetro argumento de un archivo Detallado.

V.10.10.1 PARAMETROS.

BASE>> Es un arreglo, utilizado como nombre de la base de datos desde la apertura, el cual no debe haber sido alterado, como se indicó, por el usuario.

CODIGO>> Es un arreglo conteniendo el código de seguridad de la base de datos, el cual no debe haber sido alterado a partir de la apertura de la base de datos.

CARTUCHO>> Es un arreglo conteniendo la porción de disco donde reside la base de datos, el cual no debe haber sido alterado a partir de la apertura de la base de datos.

ARCHIVO>> Es una variable conteniendo el nombre del archivo de la base de datos, del cual deseamos leer y deberá ser una cadena de 6 caracteres alfanumericos.

CAMPO>> Es una arreglo conteniendo el nombre del campo llave del archivo que estamos accesando y deberá ser una cadena de 6 caracteres alfanuméricos.

BUFFER>> Es un arreglo en el cual se enviará a la Base de Datos la información del campo que se desea borrar.

ESTADO>> Es una variable donde BORBDD regresa un número indicando si se cometió algún error (Estado <> 0), o si el llamado fue Exitoso (Estado = 0).

ARGUMENTO>> Es una variable donde se envía un número entero indicando el número de registro en el archivo de detalle que se va a borrar.

Por lo que antes se deberá haber leído el registro con un llamado LEEBDD, y tomado el valor regresado en su ARGUMENTO

V.10.10.2 ARCHIVOS DETALLADOS.

El manejador hará todos los cambios requeridos a las cadenas, incluyendo las cabezas de éstas en los archivos maestros relacionados, si el último eslabon de una cadena es borrado en un detalle conectado con un archivo maestro automático, BORBDD borrará la llave relacionada en este.

V.10.11 REPORTE DE ERRORES DE LA BASE DE DATOS.

REPER(ESTADO)

Esta rutina envia un mensaje de error a la pantalla, describiendo el error cometido al accesar la base de datos, al usar alguna de las rutinas del manejador.

V.10.11.1 PARAMETRO.

ESTADO>> Es número entero representando un error el cual es recibido de las rutinas:

1 > ABRBD
2 > CIEBD
3 > LEEBD
4 > LEEBDD
5 > PONBD
6 > PONBDD
7 > BORBD
8 > BORBDD
9 > ACTBD
10> ACTBDD

en su respectivo patrametro estado.

V.10.12 CERRADO DE LA BASE DE DATOS.

CIEBD(BASE, CODIGO, CARTUCHO, ESTADO)

Termina el acceso a la base de datos.

V.11.2.1 PARAMETROS.

BASE>> Es un arreglo que utilizado como nombre de la base de datos desde la apertura el cual no debe haber sido alterado como se indico por el usuario.

CODIGO>> Es un arreglo conteniendo el código de seguridad de la base de datos, el cual no debe haber sido alterado a partir de la apertura de la base de datos.

CARTUCHO>> Es un arreglo conteniendo la porción de disco donde reside la base de datos, el cual no debe haber sido alterado a partir de la apertura de la base de datos.

ESTADO>> Es una variable donde CIEBD regresa un numéro indicando si se cometió algún error (Estado <> 0) o si el llamado fue Exitoso (Estado = 0).

TEMA VI MODIFICACIONES para PORTABILIDAD

VI.1 JUSTIFICACION.

Uno de los objetivos principales de ésta Tesis es el de poder portar el sistema desarrollado (Manejador de Bases de Datos) en una HP-1000 a otras máquinas. Por lo que se decidió portar el sistema del Pascal 1000 al Pascal/MT+ de Digital Research.

Este capítulo tiene como objetivo el de mostrar todos los cambios que fueron necesarios para éste fin, y así demostrar la capacidad de portabilidad del manejador.

Para éstos fines fue que se eligió el lenguaje PASCAL, por su estandarización y facilidades para estructurar el desarrollo de los sistemas.

VI.2 CAMBIOS NECESARIOS.

En los siguientes incisos se enumerarán todos los cambios que fueron necesarios para lograr portar el sistema de la HP-1000 al Pascal/MT+

VI.3 CAMBIOS A NIVEL LENGUAJE.

Los cambios que hubo que hacer fueron a nivel lenguaje, y dentro de éstos cambios se pueden definir tres grupos, cambios debidos al manejo de archivos directos, cambios debidos a conceptos del lenguaje y cambios debido al manejo de archivos secuenciales.

VI.3.1 CAMBIOS DEBIDOS A CONCEPTO DE LENGUAJE

Dentro de éste grupo tenemos dos diferencias entre el pascal de la HP-1000 y el Pascal/MT+.

A) FORMA DE COMENTAR.

En el pascal de la HP-1000 los comentarios pueden ir entre llaves '{ comen }' o entre paréntesis y asteriscos '(* comen *)' y en el Pascal/MT+ solo se permiten comentarios del tipo '(* comen *)'.

para éstos fines se cambiaron todos los comentarios al formato aceptado por el Pascal/MT+.

B) SUBPROGRAMAS.

La forma de declarar un subprograma en el Pascal de la HP-1000 difiere de la forma en que se hace en el Pascal/MT+.

Dentro de la HP-1000 se utiliza la declaración EXTERNAL para indicar que la rutina llamada es un subprograma de la siguiente manera.

>>> PROCEDURE XXX (PARAMETROS) ; EXTERNAL

Mientras que en el Pascal/MT+ se declará.

>>> EXTERNAL PROCEDURE XXX (Parámetros);

En cuanto a la estructura del subprograma dentro del Pascal de la HP-1000 es la siguiente:

>>> $SUBPROGRAM

>>> PROGRAM XXX

>>> CUERPO DEL SUBPROGRAMA (Programa tipo Pascal)

>>> .

Y dentro del Pascal/MT+ es la siguiente.

>>> MODULE XXX;

>>> CUERPO DEL SUBPROGRAMA (Programa tipo Pascal)

>>> MOEND.

VI.3.2 CAMBIOS DEBIDOS AL MANEJO DE ARCHIVOS DIRECTOS.

La forma en que se manejó la lectura de archivos directos en la HP-1000 fue con un grupo de intrinsecos

(Llamados FMP)

>>> Fmpopen para abrir el archivo

>>> Fmpclose para cerrar el archivo

>>> Fmpsetposition para posicionarse en un registro del archivo

>>> Fmpread para leer del archivo

>>>Fmpwrite para escribir en el del archivo

y se englobaron en 4 rutinas para manejo de archivos directos.

>>> OPEND para abrir el Archivo

>>> CLOSED para cerrar el archivo

>>> WRITED para escribir en el archivo

>>> READD para leer del archivo

en el PASCAL/MT+ se sustituyeron éstas rutinas e intrínsecos por las siguientes llamadas de Pascal.

>>> OPEN para abrir el archivo y asignarlo a un Archivo físico en disco

>>> CLOSE para cerrar el archivo

>>> SEEKREAD para posicionarse en el Archivo y leer datos

>>> SEEKWRITE para posicionarse en el Archivo y escribir datos.

VI.3.3 CAMBIOS DEBIDOS AL MANEJO DE ARCHIVOS SECUENCIALES.

El archivo Raíz de la Base de Datos del cual hemos hablado en capítulos anteriores se trata de un archivo

secuencial, por lo que se tuvo que observar la diferencia de manejo de éstos archivos entre el Pascal 1000 y el Pascal/MT+ .

Dentro de la HP-1000 la forma de manejarlos es la siguiente:

>>> RESET (Nombre lógico,Nombre físico); {Abre el archivo para lectura}

o

>>> REWRITE (Nombre lógico,Nombre físico); {Abre el archivo para escritura}

y entonces se puede escribir o leer del archivo.

>>> READ(Nombre lógico,formato); {para leer }

>>> WRITE(Nombre lógico,formato); {para Escribir}

En el Pascal/MT+ ésto cambia ligeramente.

>>> ASSIGN (Nombre lógico,Nombre físico);

y entonces

>>> RESET (Nombre lógico); {Abre el archivo para lectura}

o

>>> REWRITE (Nombre lógico); {Abre el archivo para escritura}

y entonces se puede escribir o leer del archivo.

>>> READ(Nombre lógico,formato); {para leer }

>>> WRITE(Nombre lógico,formato); {para Escribir}

VI.4 TABLA DE CAMBIOS EFECTUADOS.

En la siguiente tabla se mostrarán cada uno de los programas y cada una de las instrucciones que fueron cambiadas, las intersecciones nos muestran el número de veces que tuvo que ser cambiada una instrucción por otra equivalente, y en la última columna y último renglón se muestran los totales.

	INSTRUCCION								
	SEEK READ	SEEK WRITE	MODULE	OPEN	CLOSE	ASSI-NG	RESET	REWRI-TE	EXTER-NAL	TOTAL xPROG
ABRE P 200	0	0	2	0	0	2	1	1	0	6
ACT R 370	2	1	2	1	1	1	1	0	0	9
ACTD O 334	0	1	2	1	1	1	1	0	0	7
BOR G 415	3	4	2	1	1	1	1	0	0	13
BORD R 578	11	12	2	2	9	1	2	0	1	40
CIERRA A 197	0	0	2	0	0	2	1	1	0	6
ESC M 491	5	2	2	1	2	1	1	0	0	14
ESCD A 639	10	4	2	3	3	1	1	0	3	27
GENBA 1004	0	2	0	1	1	4	10	3	0	21
GENBA2 1013	0	2	0	1	1	4	10	3	0	21
LEE 391	2	0	2	1	1	1	1	0	0	8
LEED 480	3	0	2	2	2	1	1	0	0	11
REPER 119	0	0	2	0	0	0	0	0	0	2
TOTAL IX INST 6231	36	28	22	14	22	20	31	8	4	185

VI.5 CONCLUSION

De la observación de la tabla se obtienen los siguientes resultados

Número total de líneas = 6231

Número de líneas modificadas = 185

Porcentaje de líneas modificadas respecto al total = 2%

De los cambios vistos se puede concluir que no es muy laborioso lograr la portabilidad de éste sistema, lo que hace se cumpla uno de los objetivos de ésta tesis, que es la portabilidad del manejador.

APENDICE I

para facilitar el manejo de las rutinas de la Base de Datos y poder interpretar los posibles errores al accesarla, se creo un código de errores del 0 al 18.

Código		Explicación
0	--->	No ocurrio ningún error
1	--->	Archivo Maestro Lleno
2	--->	Llave Existente
3	--->	Nombre de Base erróneo
4	--->	Base de datos abierta previamente
5	--->	Base de datos cerrada previamente
6	--->	Nombre de Base la base no se cerró
7	--->	La base de datos está cerrada
8	--->	El Archivo referido no existe
9	--->	Campo llave no existe en Raíz
10	--->	Se trata de un archivo automático
11	--->	La llave a actualizar no existe
12	--->	La llave a leer no existe
13	--->	La llave a Borrar no existe
14	--->	Llave no existe en maestro manual
15	--->	Archivo de detalle lleno
16	--->	Archivo referenciado no detallado
17	--->	Fin de la cadena
18	--->	La Cadena Está Vacía

Estos mensajes se podrán desplegar desde programa con la rutina de la base REPER .

APENDICE II

AII.1 PROGRAMA DE APLICACION

```
$DEBUG,AUTOPAGE,STATS,RECURSIVE ON,WARN OFF,CODE_INFO ON
PROGRAM LLAMADO(Input,output);
{ ********************************************************************** }
{ *                                                                    * }
{ *  Este programa prueba los intrinsecos del manejador de Bases de    * }
{ *  Datos.                                                            * }
{ *                                                                    * }
{ *                                                                    * }
{ *  PROGRAMA              :   LLAMAD                                  * }
{ *                                                                    * }
{ *  LENGUAJE              :   PASCAL                                  * }
{ *                                                                    * }
{ *  ESCRITO               :   SEP/85                                  * }
{ *                                                                    * }
{ *  DISENO CREADO  POR    :   Joaquin Tomas Merono.                   * }
{ *                                                                    * }
{ *  Este programa es parte del software del manejador de bases        * }
{ *  de datos.                                                         * }
{ *                                                                    * }
{ ********************************************************************** }
```

```
{ ************************************************************************ }
{ *                                                                      * }
{ *            DECLARACION DE VARIABLES Y TIPOS GLOBALES                 * }
{ *                                                                      * }
{ ************************************************************************ }
TYPE
   Entero     = -32767..32767                    ;
   Regtab_Ent = -32767..32767                    ;
   Regtab_Asc = PACKED ARRAY [1..2] OF CHAR      ;
   Nombre     = PACKED ARRAY [1..6] OF CHAR      ;
   Dato       = PACKED ARRAY [1..250] OF CHAR    ;
   Re = RECORD
         Ma  : Entero ;
         Aa  : Entero ;
         Ap  : Entero ;
         D1  : Entero ;
         D2  : Entero ;
         D3  : Entero ;
         LLA : PACKED ARRAY [1..8] OF CHAR  ;
         DA1 : PACKED ARRAY [1..4] OF CHAR  ;
         DA2 : PACKED ARRAY [1..30] OF CHAR ;
         DA3 : REAL                         ;
         DA4 : INTEGER                      ;
   END;


VAR
   Nomba          : Nombre                    ; { Nombre del Archivo Raiz   }
   Nomar          : Nombre                    ; { Nombre del Archivo        }
   Nomlla         : Nombre                    ; { Nombre de la llave        }
   Codigo         : Regtab_Asc                ; { Codigo de seguridad       }
   Cartucho       : Regtab_Asc                ; { Cartucho del Archivo Raiz }
   Estado         : Entero                    ; { CODIGO DE ERROR           }
   Buf            : Re                        ; { Buffer a transmitir       }
   XX             : DATO                      ; { Buffer a transmitir       }
   Opcion         : INTEGER                   ; { Para seleccionar funcion  }
   Esc            : CHAR                      ;

{ ************************************************************************** }
{ *                                                                        * }
{ *  Procedimiento que revisa la validez de el llamado para dar de alta,   * }
{ *  y en base del tipo de archivo de que se trate elige la rutina ade-    * }
{ *  cuada para escribir los datos dentro de los archivos de la Base de    * }
{ *  datos.                                                                * }
{ *                                                                        * }
{ ************************************************************************** }
PROCEDURE PONBDD(Nomba:Nombre; Codigo,Cartucho:Regtab_Asc;
                 Nomar,Nomlla:Nombre; Regis : RE; Var Estado:Entero);EXTERNAL;
```

```
{ ************************************************************************ }
{ *                                                                      * }
{ *  Procedimiento que revisa la validez de el llamado para dar de Leer, * }
{ *  datos en archivos de detalle.                                       * }
{ *                                                                      * }
{ ************************************************************************ }
PROCEDURE LEEBDD(Nomba:Nombre; Codigo,Cartucho:Regtab_Asc;
   Nomar,Nomlla:Nombre;Var Regis : RE; Var Estado,Opci,Pos:Entero);EXTERNAL;
{ ************************************************************************ }
{ *                                                                      * }
{ *  Procedimiento que revisa la validez de el llamado para Borrar ,     * }
{ *  datos en archivos de detalle.                                       * }
{ *                                                                      * }
{ ************************************************************************ }
PROCEDURE BORBDD(Nomba:Nombre; Codigo,Cartucho:Regtab_Asc;
   Nomar,Nomlla:Nombre;Var Regis : RE; Var Estado,Opci,Pos:Entero);EXTERNAL;
{ ************************************************************************ }
{ *                                                                      * }
{ *  Procedimiento que revisa la validez de el llamado para Actualizar , * }
{ *  datos en archivos de detalle.                                       * }
{ *                                                                      * }
{ ************************************************************************ }
PROCEDURE ACTBDD(Nomba:Nombre; Codigo,Cartucho:Regtab_Asc;
  Nomar,Nomlla:Nombre;Var Regis : RE; Var Estado:Entero;Pos:Entero);EXTERNAL;
{ ************************************************************************ }
{ *                                                                      * }
{ *  Procedimiento para revisar la validez de la apertura de la base     * }
{ *                                                                      * }
{ ************************************************************************ }
PROCEDURE Abrbd (Nomba:Nombre;Codigo,Cartucho:Regtab_Asc;Var Estado:Entero);
EXTERNAL;
{ ************************************************************************ }
{ *                                                                      * }
{ *  Procedimiento para revisar la validez del cerrado                   * }
{ *                                                                      * }
{ ************************************************************************ }
PROCEDURE Ciebd(Nomba:Nombre;Codigo,Cartucho:Regtab_Asc;Var Estado:Entero);
EXTERNAL;
```

```
{ ************************************************************************ }
{ *                                                                      * }
{ *  Procedimiento para revisar la validez del cerrado                   * }
{ *                                                                      * }
{ ************************************************************************ }
PROCEDURE Reper(Estado:Entero);EXTERNAL;
{ ************************************************************************ }
{ *                                                                      * }
{ *  Procedimiento para Programar las tecals en el menu de entrada       * }
{ *                                                                      * }
{ ************************************************************************ }
PROCEDURE P1;
BEGIN
     Esc := CHR(27);
     WRITELN (Esc,'&f2a1k16d20LALTAS          1                    ');
     WRITELN (Esc,'&f2a2k16d20LBAJAS          2                    ');
     WRITELN (Esc,'&f2a3k16d20LCAMBIOS        3                    ');
     WRITELN (Esc,'&f2a4k16d20LREPORTE        4                    ');
     WRITELN (Esc,'&f2a5k16d20LFIN            5                    ');
     WRITELN (Esc,'&f2a6k16d20L               5                    ');
     WRITELN (Esc,'&f2a7k16d20L               5                    ');
     WRITELN (Esc,'&f2a9k16d20L               5                    ');
     WRITELN (Esc,'&jB');
END;
{ ************************************************************************ }
{ *                                                                      * }
{ *  Procedimiento para Desplegar el menu de opciones                    * }
{ *                                                                      * }
{ ************************************************************************ }
PROCEDURE Menu;
BEGIN
     Esc := CHR(27);
     WRITELN (Esc,'h',Esc,'J');
     WRITELN (Esc,'&a2r20C*****************************************');
     WRITELN (Esc,'&a3r20C*                                       *');
     WRITELN (Esc,'&a4r20C*      PROGRAMA PILOTO PARA PROBAR      *');
     WRITELN (Esc,'&a5r20C*                                       *');
     WRITELN (Esc,'&a6r20C*    INTRINSECOS DE LA BASE DE DATOS    *');
     WRITELN (Esc,'&a7r20C*                                       *');
     WRITELN (Esc,'&a8r20C*              (A,B,C,R)                *');
     WRITELN (Esc,'&a9r20C*                                       *');
     WRITELN (Esc,'&a10r20C*        EN ARCHIVOS DETALLADOS        *');
     WRITELN (Esc,'&a11r20C*                                      *');
     WRITELN (Esc,'&a12r20C*****************************************');
     WRITELN (Esc,'&a15r20C          Altas                        ');
     WRITELN (Esc,'&a16r20C          Bajas                        ');
     WRITELN (Esc,'&a17r20C          Cambios                      ');
     WRITELN (Esc,'&a18r20C          Reporte                      ');
     WRITELN (Esc,'&a19r20C          Fin                          ');
     PROMPT  (Esc,'&a22r20C          OPCION ?  ');
END;
```

```
{ ************************************************************************** }
{ *                                                                        * }
{ *  Procedimiento Actualizar en Archivos de Detalle con 'ACTD'            * }
{ *                                                                        * }
{ ************************************************************************** }
PROCEDURE Cambios2;
VAR
   Estad : INTEGER;
   Opci  : Entero ;
   Opcion: PACKED ARRAY ¡1..2¿ OF CHAR;
   Pos   : Entero ;
   Num   : Entero ;
   DUMMY : CHAR   ;
BEGIN
     Esc := CHR(27);
     WRITELN (Esc,'h',Esc,'J');

     PROMPT (Esc,'&a7r20CIngresa la clave (. Para fin) ..... ? ',Esc,'K');
     READLN (BUF.LLA);
     Opci   := 0        ;
     IF BUF.LLA¡1¿ <> '.' THEN
         WHILE Estad <> 17 DO
         BEGIN
               Nomlla := 'EMPLED' ;
               Nomar  := 'ARCOMP' ;
               Leebdd(Nomba,Codigo,Cartucho,Nomar,Nomlla,Buf,Estado,Opci,Pos);
               WRITELN(Esc,'&a10r20C1) Departamento..: ',BUF.DA1);
               WRITELN(Esc,'&a11r20C2) Puesto.......: ',BUF.DA2);
               WRITELN(Esc,'&a12r20C3) Sueldo.......: ',BUF.DA3:9:2);
               WRITELN(Esc,'&a13r20C4) Antiguedad....: ',BUF.DA4);
               IF (ESTADO <> 0) AND (ESTADO <> 17) THEN
               BEGIN
                  REPER (Estado);
                  Estad :=17
               END(IF);
               IF ESTADO = 17 THEN
               BEGIN
                   REPER(Estado);
                   Estad := 17 ;
               END(IF);
               IF Estad <> 18 THEN BEGIN
               PROMPT (Esc,'&a22r20CQUIERES ACTUALIZAR EL REGISTRO ? ');
               READLN (OPCION);
               IF OPCION <> 'SI' THEN OPCION := 'NO' ;
               WHILE OPCION = 'SI' DO
               BEGIN
                  PROMPT(Esc,'&a22r20CELIGE EL NUMERO DE OPCION  ? ',Esc,'K');
                  READLN (Num);
                  CASE Num OF
                     1 :  BEGIN
                            PROMPT (Esc,'&a22r20C1) Departamento? ',Esc,'K');
                            READLN (BUF.DA1);
                          END;
                     2 :  BEGIN
                            PROMPT (Esc,'&a22r20C2) Puesto       ? ',Esc,'K');
                            READLN (BUF.DA2);
                          END;
```

```
                    3 :  BEGIN
                           PROMPT (Esc,'&a22r20C3) Sueldo      ? ',Esc,'K');
                           READLN (BUF.DA3);
                         END;
                    4 :  BEGIN
                           PROMPT (Esc,'&a22r20C4) Antiguedad  ? ',Esc,'K');
                           READLN (BUF.DA4);
                         END;
                  END{CASE};
                  Actbdd(Nomba,Codigo,Cartucho,Nomar,Nomlla,Buf,Estado,Pos);
                  WRITELN(Esc,'&a10r20C1) Departamento...: ',BUF.DA1);
                  WRITELN(Esc,'&a11r20C2) Puesto.....: ',BUF.DA2);
                  WRITELN(Esc,'&a12r20C3) Sueldo.....: ',BUF.DA3:9:2);
                  WRITELN(Esc,'&a13r20C4) Antiguedad.....: ',BUF.DA4);
                  READ (DUMMY);
                  IF ESTADO <> 0  THEN
                  BEGIN
                     REPER (Estado);
                     Estad :=17
                  END{IF};
               PROMPT (Esc,'&a22r20CQUIERES ACTUALIZAR EL REGISTRO ? ');
               READLN (OPCION);
            END{WHILE};
            END{IF};
         END{WHILE};
      {ENDIF};
END;
{ ************************************************************************** }
{ *                                                                        * }
{ *  Procedimiento Borrar en Archivos de Detalle con 'BORBDD'              * }
{ *                                                                        * }
{ ************************************************************************** }
PROCEDURE Baja2;
VAR
   Estad : INTEGER;
   Opci  : Entero ;
   Opcion: PACKED ARRAY [1..2] OF CHAR;
   Pos   : Entero ;
   Num   : Entero ;
   DUMMY : CHAR   ;
BEGIN
     Esc := CHR(27);
     WRITELN (Esc,'h',Esc,'J');

     PROMPT (Esc,'&a7r20CIngresa la clave (. Para fin) ..... ? ',Esc,'K');
     READLN (BUF.LLA);
     Opci   := 0        ;
     IF BUF.LLA[1] <> '.' THEN
         WHILE Estad <> 17 DO
         BEGIN
            Nomlla := 'EMPLED' ;
            Nomar  := 'ARCOMP' ;
```

```
            Leebdd(Nomba,Codigo,Cartucho,Nomar,Nomlla,Buf,Estado,Opci,Pos);
            IF Estado <> 18 THEN
            BEGIN
               WRITELN(Esc,'&a10r20C1) Departamento..: ',BUF.DA1);
               WRITELN(Esc,'&a11r20C2) Puesto.....: ',BUF.DA2);
               WRITELN(Esc,'&a12r20C3) Sueldo.....: ',BUF.DA3:9:2);
               WRITELN(Esc,'&a13r20C4) Antiguedad.....: ',BUF.DA4);
            END(IF);
            IF (ESTADO <> 0) AND (ESTADO <> 17) THEN
            BEGIN
               REPER (Estado);
               Estad :=17
            END(IF);
            IF ESTADO = 17 THEN
            BEGIN
                REPER(Estado);
                Estad := 17 ;
            END(IF);
            PROMPT (Esc,'&a22r20CQUIERES BORRAR EL REGISTRO ? ');
            READLN (OPCION);
            IF OPCION <> 'SI' THEN OPCION := 'NO' ;
            IF OPCION = 'SI' THEN
            BEGIN
              Borbdd(Nomba,Codigo,Cartucho,Nomar,Nomlla,Buf,Estado,Opci.Pos);
              IF ESTADO <> 0  THEN
              BEGIN
                 REPER (Estado);
                 Estad :=17
              END(IF);
            END(IF);

        END(WHILE);
    (ENDIF);
END;
{ ************************************************************************** }
{ *                                                                        * }
{ *  Procedimiento dar de alta utilizando PONBDD                           * }
{ *                                                                        * }
{ ************************************************************************** }
PROCEDURE Alta2;
BEGIN
     Esc := CHR(27);
     WRITELN (Esc,'h',Esc,'J');

     WHILE Buf.Llaji¿ <> '.'  DO
     BEGIN
         PROMPT (Esc,'&a7r20CIngresa la clave (. Para fin) ..... ? ',Esc,'K');
         READLN (BUF.LLA);
         IF BUF.LLAji¿ <> '.' THEN
         BEGIN
               PROMPT (Esc,'&a10r20CIngresa Departamento..? ',Esc,'K');
               READLN (BUF.DA1);
               PROMPT (Esc,'&a11r20CPuesto.....? ',Esc,'K');
               READLN (BUF.DA2);
               PROMPT (Esc,'&a12r20CSueldo.....? ',Esc,'K');
               READLN (BUF.DA3);
               PROMPT (Esc,'&a13r20CAntiguedad.....? ',Esc,'K');
               READLN (BUF.DA4);
```

```
                Nomlla := 'EMPLED' ;
                Nomar  := 'ARCOMP' ;
                Ponbdd(Nomba,Codigo,Cartucho,Nomar,Nomlla,Buf,Estado);
                IF ESTADO <> 0 THEN REPER(Estado);
          END(IF);
      END(WHILE);
END;


{ ************************************************************************** }
{ *                                                                        * }
{ *  Procedimiento dar de alta utilizando PONBDD                           * }
{ *                                                                        * }
{ ************************************************************************** }
PROCEDURE Report2;
VAR
   Estad : INTEGER;
   Opci  : Entero ;
   Pos   : Entero ;
   DUMMY : CHAR   ;
BEGIN
     Esc := CHR(27);
     WRITELN (Esc,'h',Esc,'J');

     PROMPT (Esc,'&a7r20CIngresa la clave (. Para fin) ..... ? ',Esc,'K');
     READLN (BUF.LLA);
     Opci   := 0        ;
     IF BUF.LLA[1] <> '.' THEN
         WHILE Estad <> 17 DO
         BEGIN
               Nomlla := 'EMPLED' ;
               Nomar  := 'ARCOMP' ;
               Leebdd(Nomba,Codigo,Cartucho,Nomar,Nomlla,Buf,Estado,Opci,Pos);
               IF Estado <> 18 THEN
               BEGIN
                  WRITELN(Esc,'&a10r20C Departamento..: ',BUF.DA1);
                  WRITELN(Esc,'&a11r20CPuesto.........: ',BUF.DA2);
                  WRITELN(Esc,'&a12r20CSueldo.........: ',BUF.DA3:9:2);
                  WRITELN(Esc,'&a13r20CAntiguedad.....: ',BUF.DA4);
               END(IF);
               IF (ESTADO <> 0) AND (ESTADO <> 17) THEN
               BEGIN
                  REPER (Estado);
                  Estad :=17
               END(IF);
               IF ESTADO = 17 THEN
               BEGIN
                   REPER(Estado);
                   Estad := 17 ;
               END(IF);
               READ (DUMMY);
          END(WHILE);
     (ENDIF);
END;
```

```
{ ************************************************************************ }
{ *                                                                      * }
{ *                  P R O G R A M A   P R I N C I P A L                 * }
{ *                                                                      * }
{ ************************************************************************ }

  BEGIN
     Nomba := 'MIBASE' ;
     Codigo:= 'JT' ;
     Cartucho:= '18' ;
     Abrbd(Nomba,Codigo,Cartucho,Estado)    ;
     IF ESTADO <> 0 THEN REPER(Estado);
     WHILE OPCION <> 5 DO
     BEGIN
        P1              ;
        Menu            ;
        READLN (Opcion) ;

        CASE Opcion OF
           1 : Alta2    ;
           2 : Baja2    ;
           3 : Cambios2;
           4 : Report2 ;
           OTHERWISE
              Opcion:= 5;
        END(CASE);
        BUF.Lla := ' ';

     END(WHILE);
     Clebd(Nomba,Codigo,Cartucho,Estado)    ;
     IF ESTADO <> 0 THEN REPER(Estado);
  END.
```

APENDICE III

AIII.1 BIBLIOGRAFIA

DESIGN OF DATA BASE STRUCTURES

TOBY J. PEDREY

JAMES P. FRY

PRENTICE HALL SOFTWARE SERIES 1982

PASCAL 1000 REFERENCE MANUAL

HEWLETT-PACKARD 1985

IMAGE 1000 DATA BASE MANAGMENT SYSTEM REFERENCE MANUAL

HEWLETT-PACKARD 1985

PASCAL/MT+ USERS MANUAL

DIGITAL RESEARCH 1982