FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

FOX PRO 2.6 BASICO PARA AMBIENTE WINDOWS

10 al 20 de junio de 1996

DIRECTORIO DE PROFESORES

ING. CLAUDIA ZAVALA DIAZ
COORDINADOR DE PROYECTO
CENTRO DE ESTUDIOS EDUCATIVOS
AV. REVOLUCION No. 1291
COL. TLACOPAC SAN ANGEL
DELEGACION ALVARO OBREGON
C.P. 01040 MEXICO, D.F.
TEL: 593 59 77 ext. 41 ó 42

TEC. L. ENRIQUE SANCHEZ GONZALEZ
COORDINADOR GENERAL
UNIDAD DE SISTEMAS
UNIVERSIDAD INTERCONTINENTAL, S.A.
INSURGENTES SUR No. 4303 PISO 2
TLALPAN, MEXICO,D.F.
TEL: 573 85 44 exts. 1037 ó 1039

'pmc.

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA, UNAM
CURSOS ABIERTOS

CURSO: **54cc FOX PRO 2.6 BASICO PARA AMBIENTE WINDOWS**
FECHA: **10 al 20 de junio de 1996**

EVALUACIÓN DEL PERSONAL DOCENTE

(ESCALA DE EVALUACIÓN: 1 A 10)

CONFERENCISTA	DOMINIO DEL TEMA	USO DE AYUDAS AUDIOVISUALES	COMUNICACIÓN CON EL ASISTENTE	PUNTUALIDAD
ING. CLAUDIA ZAVALA DIAZ				
TEC. L. ENRIQUE SANCHEZ GLEZ.				

Promedio _____

EVALUACIÓN DE LA ENSEÑANZA

CONCEPTO	CALIF.
ORGANIZACIÓN Y DESARROLLO DEL CURSO	
GRADO DE PROFUNDIDAD DEL CURSO	
ACTUALIZACIÓN DEL CURSO	
APLICACIÓN PRACTICA DEL CURSO	

Promedio _____

EVALUACIÓN DEL CURSO

CONCEPTO	CALIF.
CUMPLIMIENTO DE LOS OBJETIVOS DEL CURSO	
CONTINUIDAD EN LOS TEMAS	
CALIDAD DEL MATERIAL DIDÁCTICO UTILIZADO	

Promedio _____

Evaluación total del curso_____

Continúa...2

1. ¿Le agradó su estancia en la División de Educación Continua?

SI ☐ NO ☐

Si indica que "NO" diga porqué: _____

2. Medio a través del cual se enteró del curso:

Periódico *Excélsior*	
Periódico *La Jornada*	
Folleto anual	
Folleto del curso	
Gaceta UNAM	
Revistas técnicas	
Otro medio (Indique cuál)	

3. ¿Qué cambios sugeriría al curso para mejorarlo?

4. ¿Recomendaría el curso a otra(s) persona(s) ?

SI ☐ NO ☐

5. ¿Qué cursos sugiere que imparta la División de Educación Continua?

6. Otras sugerencias:

FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

M A T E R I A L D I D A C T I C O

F O X P R O 2.6

B A S I C O

JUNIO, 1996

Palacio de Minería Calle de Tacuba 5 Primer piso Deleg Cuauhtémoc 06000 México, D F APDO. Postal M-2285
Teléfonos: 512-8955 512-5121 521-7335 521-1987 Fax 510-0573 512-5121 521-4020 AL 26

FOXPRO 2.6 VERSIÓN PARA WINDOWS

1. Introducción a Foxpro

Foxpro es un manejador de bases de datos perteneciente a la compañía Microsoft, es uno de los compiladores más gustados por aquéllos usuarios que desarrollan sistemas en dos y windows.

Foxpro está compuesto por un intérprete (al estilo Dbase), con el cuál el usuario trabajará interactivamente, un editor aceptable, que permitirá la creación de archivos diversos y un compilador capaz de traducir los mandatos a lenguaje máquina (archivo obj). Todo lo anterior se tiene en el programa base, para la creación de archivos ejecutables se deberá de contar además con un kit para la construcción de éstos. Los programas ligados con el kit serán archivos de tipo ejecutable desde dos o windows dependiendo de la versión del paquete utilizado. Cabe aclarar que tanto el kit como el paquete deberán de ser para la misma versión y el mismo sistema operativo.

Consta de un sinnúmero de funciones y comandos propios del compilador, librerías y utilerías que ayudarán al desarrollador a que su tarea sea más sencilla.

La ventaja de Foxpro en comparación con otros compiladores en cuanto a bases de datos es su potencia, versatilidad y portabilidad, además de la velocidad de proceso.

Cualquier usuario que ha estado en contacto con otro tipo de manejadores encontrará fácil de aprender y manejar este compilador, ya que la interface gráfica que Foxpro maneja, la ayuda de diversos menús y su amigable ayuda serán de gran utilidad al usuario.

1.2.2 Barra de menú

La barra inmediata inferior es una barra de menú dinámica esto es, cambiará según lo que se esté realizando, aumentando o disminuyendo opciones dentro de él. Explicaremos con mayor detenimiento esta barra posteriormente.

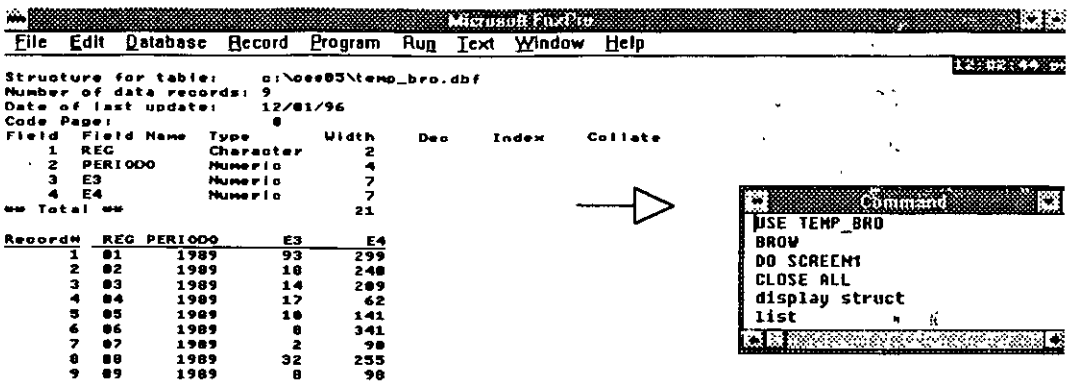
1.2.3 Ventana de command

Al entrar a Foxpro inicialmente se nos mostrará el logo del paquete en el cuerpo de la pantalla y una pequeña ventana a la que denominaremos ventana de *Command*. Esta ventana será el medio de comunicación en línea con el compilador. En esta ventana teclearemos las instrucciones u órdenes a traducir y realizar por el intérprete.

Si se ha ejecutado alguna tarea por medio de los menús, en esta ventana se desplegará automáticamente la instrucción que debió de haberse tecleado. Esto ayudará a que el usuario se familiarice con ellas, de tal manera que las utilice sin ayuda del menú, que a veces resulta ser mas rápido.

En la ventana de *command* se guardará una historia de las instrucciones tecleadas por el usuario durante su sesión de trabajo, por lo que, para ejecutar nuevamente la instrucción bastará buscarla y oprimir <enter>.

A continuación se muestra como por medio la ventana de *command* se le dan indicaciones que mostrará en el cuerpo de la pantalla, desapareciendo el logo que aparece al iniciar la aplicación.



1.2.4 Barra de estado

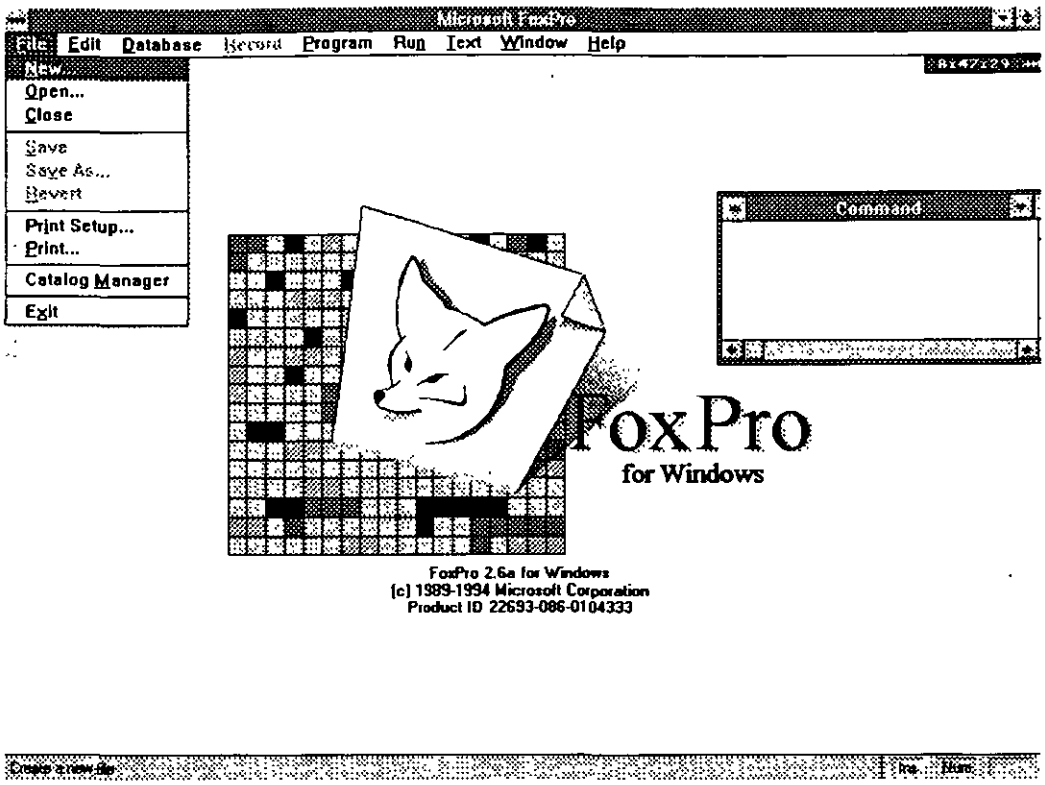
En la parte inferior del menú tendremos una barra conocida como *Barra de estado* donde el compilador nos indicará lo que está realizando, la base que se encuentra abierta, el número de registros que posee. Además el encendido y apagado de los indicadores de insertar o sobreescribir y si el teclado anexo se encuentra en forma o no numérica.

1.3 Menús

Iniciaremos con la barra de menú describiendo cada una de las opciones y su aplicación.

1.3.1 Menú Archivo (File)

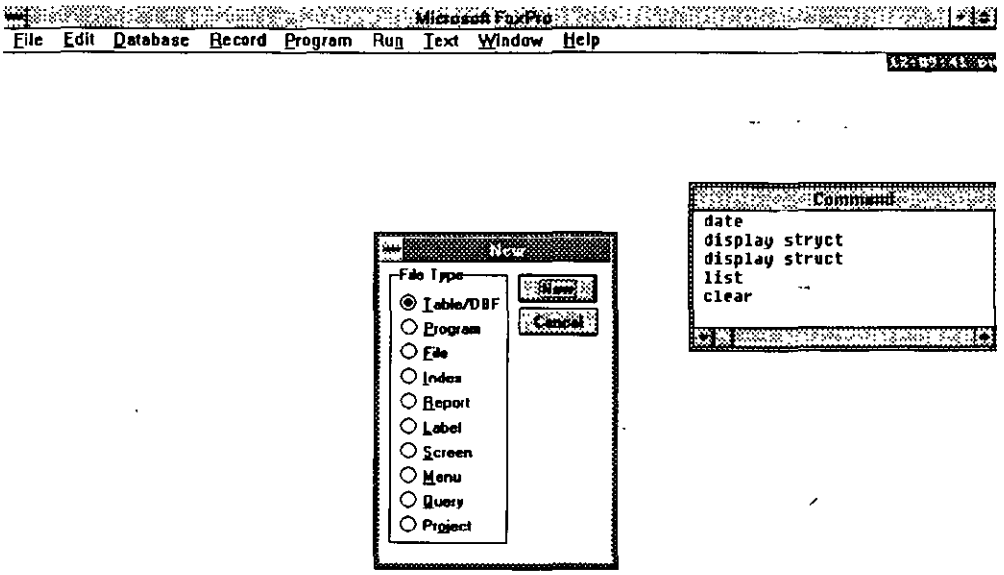
Este menú se utiliza para crear archivos, abrir, cerrar salvar, imprimir, salir de Foxpro.



1.3.1.1 Nuevo (New)

Dentro de la opción de nuevo se tendrá un menú a elegir, dentro de él será posible crear bases de datos, índices, programas, archivos ascii, reportes etiquetas, pantallas menús, querys y proyectos.

Cada una de estas opciones se tratará de forma independiente el tema que corresponda, lo importante es conocer cuál es el menú y la opción que nos permite crear diferentes tipos de aplicaciones.

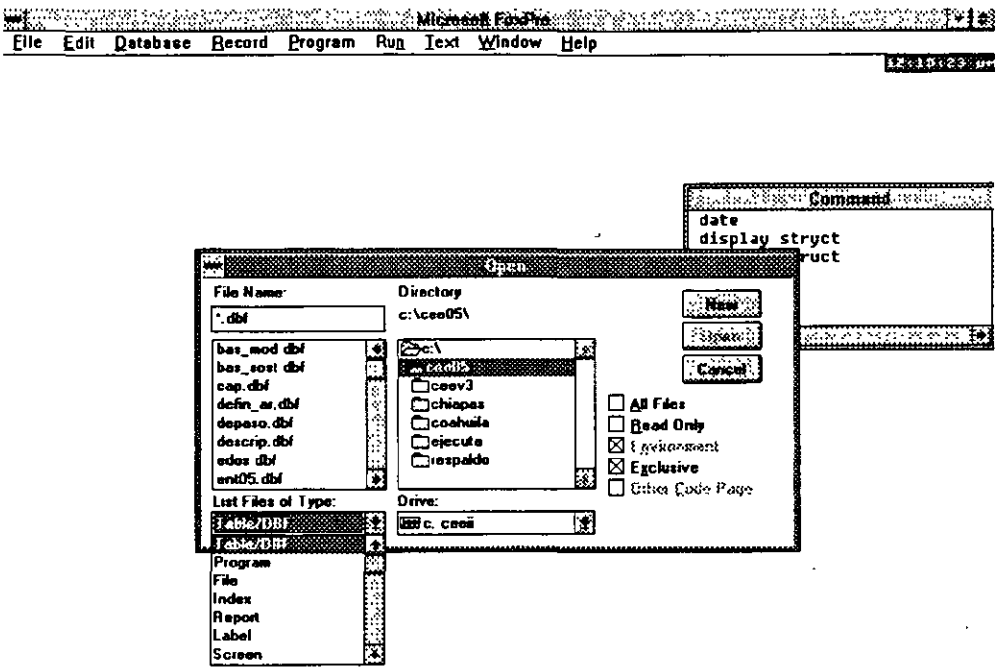


1.3.1.2 Abrir (Open)

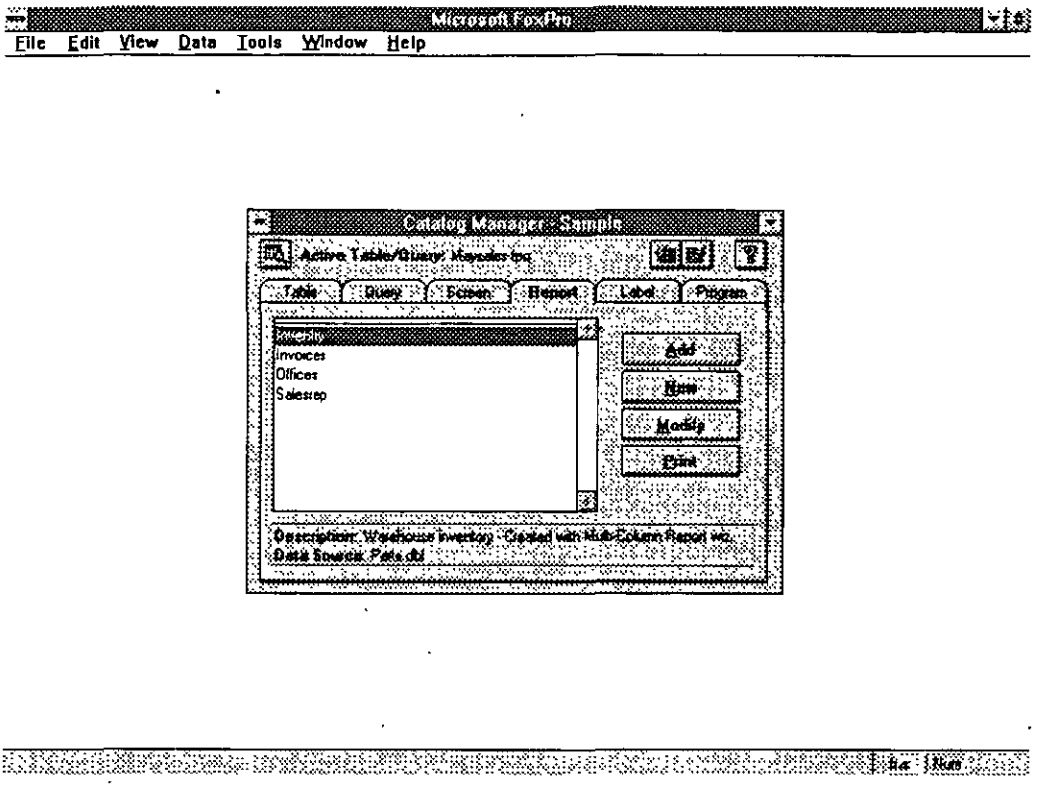
En este parte nos mostrará una pantalla clásica de windows en cuanto a la forma de selección.
Donde se tienen dos cajas de diálogo, la primera pedirá el nombre del archivo y la segunda el directorio fuente. En la parte derecha encontraremos opciones de como abrir el archivo, seleccionaremos las que consideremos necesarias. Se tiene las siguientes opciones:
Abrir todos los archivos, abrir como sólo lectura, abrir su ambiente, abrir de forma exclusiva (cuando se trata de bases de datos), con otro código de página.

Debajo de las dos cajas de diálogo encontraremos una lista desplegable, la primera mostrará las opciones de tipos de archivos que se podrán abrir (bases de datos, índices, programas, archivos, reportes, etc.).
En la caja de directorio tenemos como en cualquier aplicación windows las opciones para manejar diversas unidades de disco, con sus respectivos directorios.

Una vez hecha la selección del archivo a abrir oprimiremos cualquiera de los dos botones que se encuentran en la parte derecha de la pantalla (OK y Cancel) donde aceptamos o abortamos la acción.



Close: Cierra una base de datos, pantalla, ventana, etc.
Save: Salva la aplicación
Save as: Salva la aplicación con un nombre determinado.
Revert: Elimina las últimas modificaciones hechas a un archivo.
Print set up: Variables de inicialización para la impresión.
Print: Imprimir
Exit: Salir del paquete.
Catalogo manager: Utilería de Foxpro que nos permite manipular el paquete desde una aplicación diferente. Esta aplicación nos mostrará la pantalla siguiente:



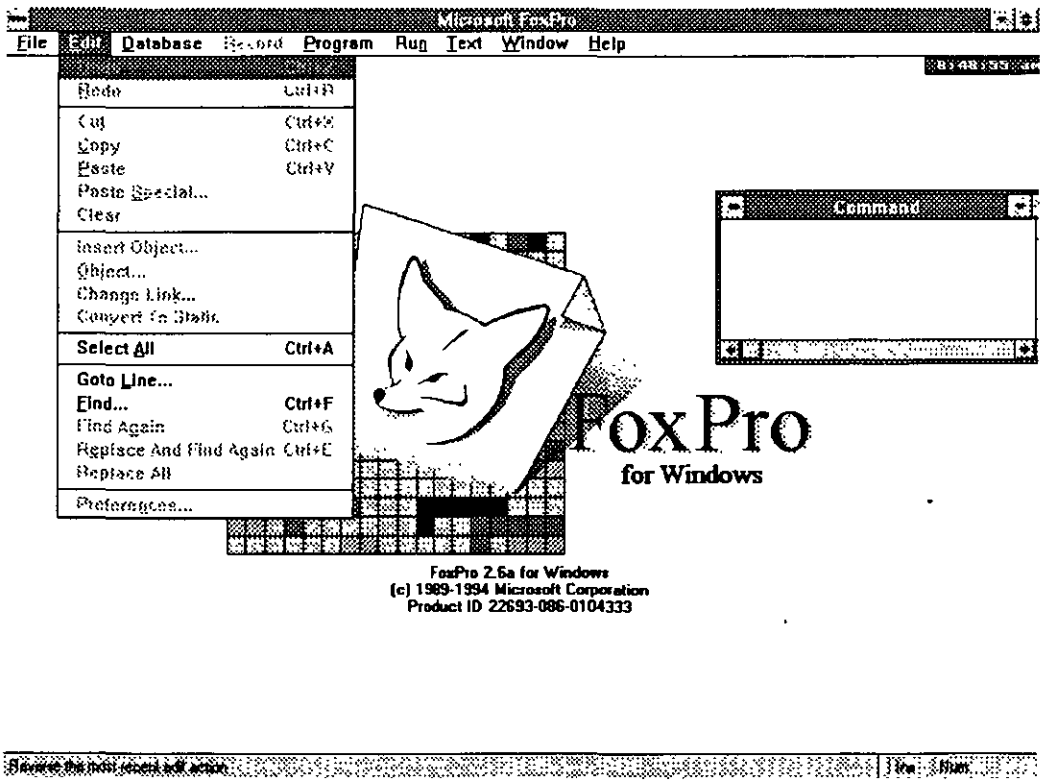
1.3.2 Menú de edición (Edit)

El menú de edición nos permitirá crear y manipular los programas hechos por el usuario.

Undo: Deshace lo último que se hizo en el editor.
Redo: Rehacer lo último hecho.
Cut: Corta un párrafo marcado.
Copy: Marca para copiar un párrafo.
Paste: Pega un párrafo marcado.
Paste special: Pega algún objeto en especial.
Clear: Borra el texto seleccionado.
Insert object: Inserta un objeto imagen a un campo de tipo general.
Object: Activa un objeto de tipo OLE.
Change link: Despliega o crea una nueva liga entre objetos.
Convert to static: Convierte a imagen un objeto OLE
Select all: Selecciona todo el texto.
Goto line: Mueve el cursor a una línea especifica.

Find: Busca un valor, cadena de caracteres, caracter especial, dentro del texto.
Find again: Busca nuevamente lo solicitado con Find.
Replace and find again: Busca y sustituye por un valor que se le determine.
Replace all: Reemplaza todas las ocurrencias.
Preferences: Establece defaults para el editor.

Estas opciones sólo estarán disponibles si nos encontramos dentro del editor.



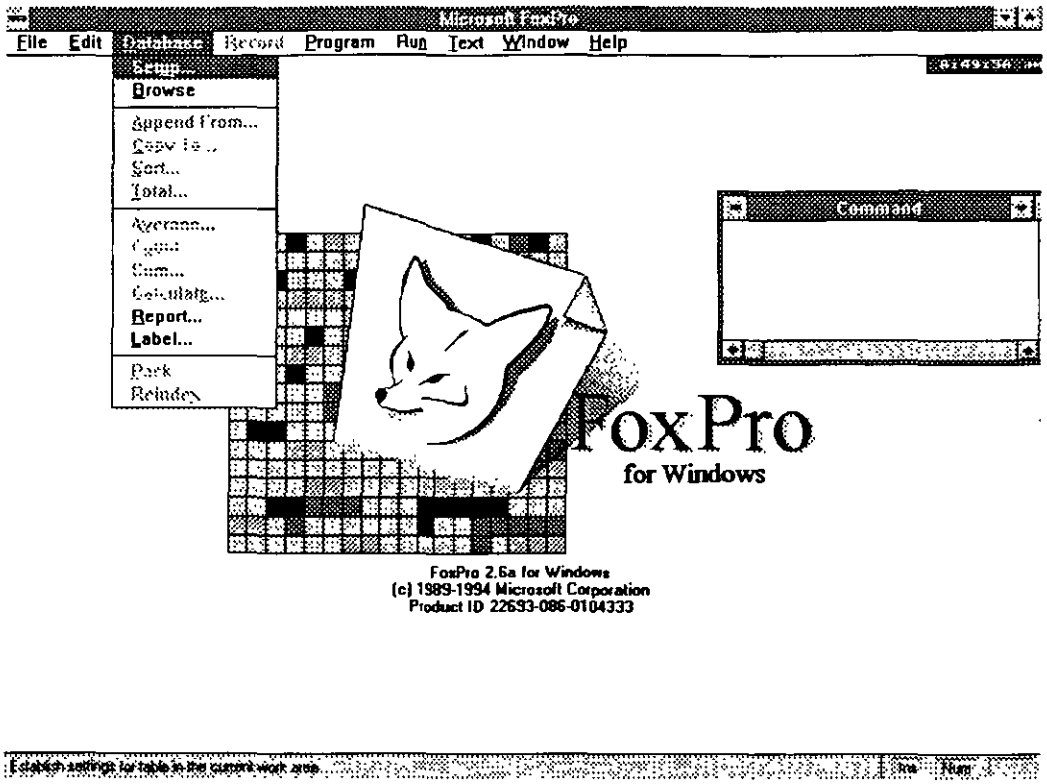
1.3.3 Menú para el manejo de la base de datos (Database)

Por medio de este menú manipularemos a la base de datos, cabe aclarar que activarán sus opciones mientras se encuentre abierta alguna.

Este menú es importante porque aquí se encuentra la opción para poder modificar la estructura de la base de datos y los índices.

Setup: Permite modificar la base de datos y los índices.
Browse: Despliega el contenido de la base de datos.
Append from: Agrega datos a una base de datos de otra ya existente.
Copy to: Copia el contenido de una base de datos a un archivo con un formato específico (ascii, wk1,wks, etc.)
Sort: Ordena el contenido de la base de datos, dejando el resultado en otra.
Total: Obtiene el total de algún campo en específico.
Average: Promedio de alguna variable numérica seleccionada.
Count: Cuenta los elementos con alguna característica en especial que existan en la base de datos.
Sum: Suma el contenido del campo determinado.
Calculate: Obtiene valores estadísticos de campos numéricos existentes en la base de datos.
Report: Envía a un reporte el contenido de la base de datos.
Label: Crea etiquetas a partir del contenido de una base de datos.
Pack: Borra los registros marcados para ser borrados.
Reindex: Reindexa una base de datos.

Cada tópico se verá ampliamente en su oportunidad.



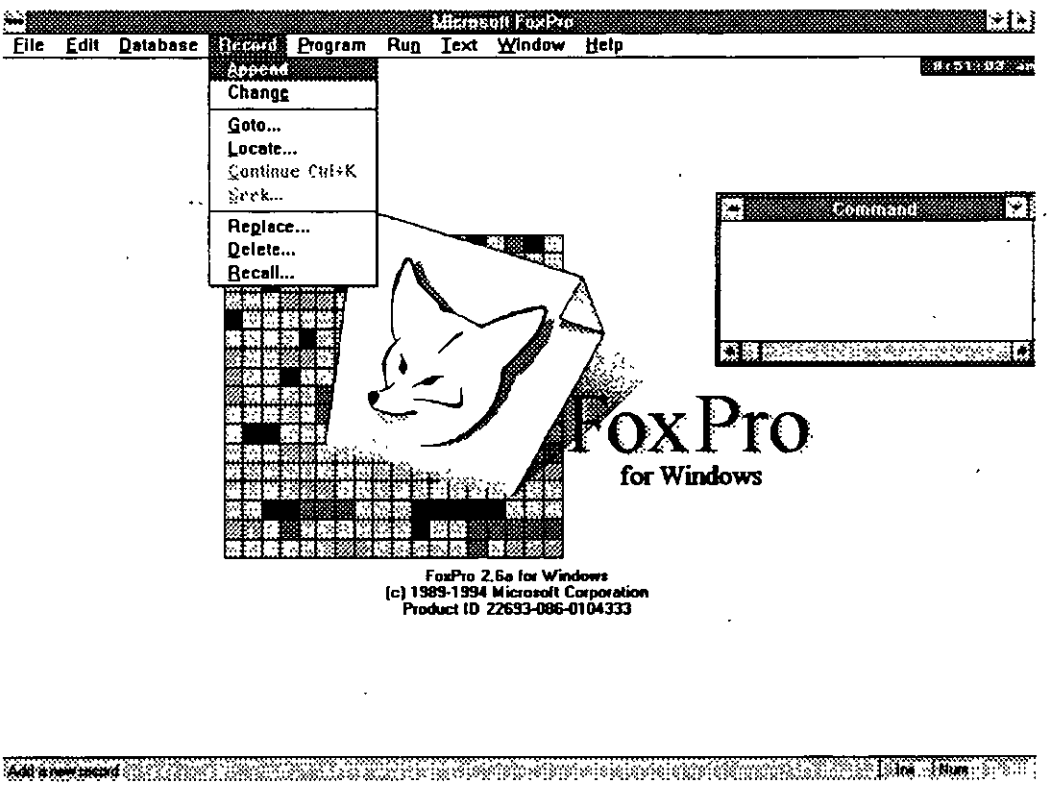
1.3.4 Menú para el manejo de registros dentro de la base de datos (Record)

El siguiente menú nos muestra todas las opciones para lo que será la manipulación de los registros existentes en la base de datos.

Append: Agrega registros a la base de datos, a partir del último existente en la base de datos.
Change: En el caso de que para agregar datos en la base se utilizara el comando *append* no nos permitirá tener acceso a la información existente anteriormente (a diferencia de dBase que si lo permite), por lo que la opción *Change* lo que hace es darle al usuario la facilidad de llamar a los registros anteriores dando opción a modificarlos.
Goto: Desplaza el apuntador de la base de datos a un registro específico.
Locate: Localiza un dato en especial y se detendrá en la primera ocurrencia.
Continue: Continúa la búsqueda iniciada con locate.

Seek: Busca una clave en especial en una base indexada.
Replace: Reemplaza el contenido de un registro por otro.
Delete: Marca registros para ser borrados.
Recall: Elimina las marcas de borrado de los registros que las posean.

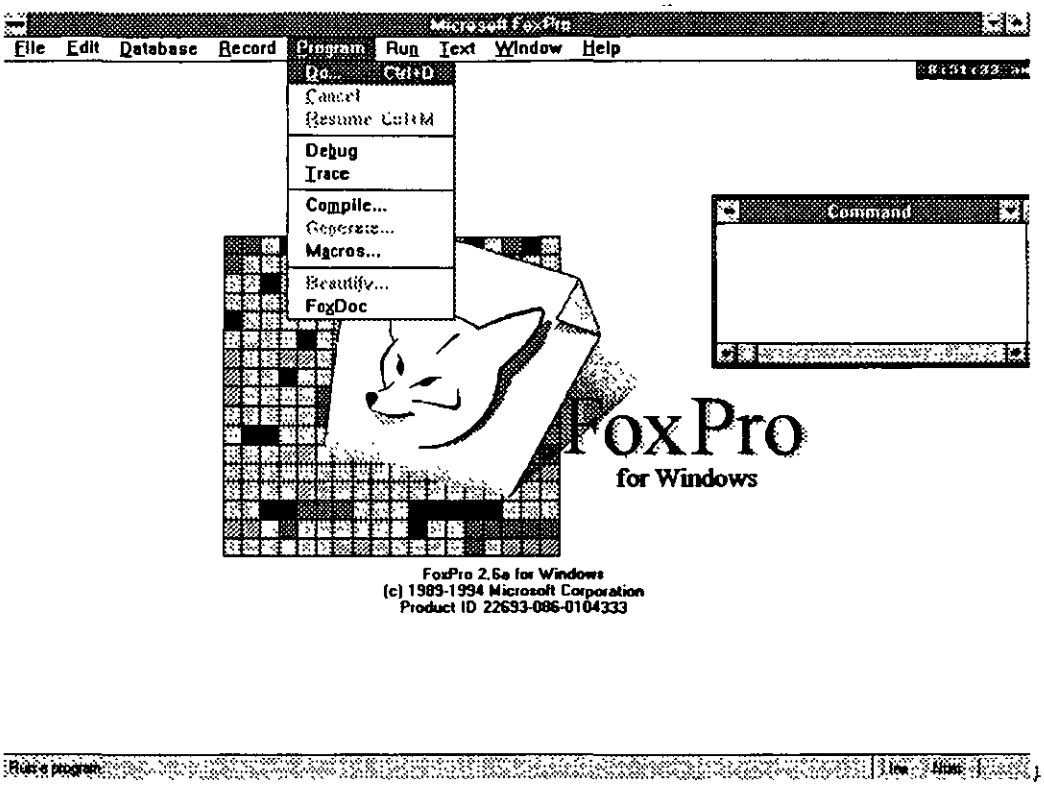
Veremos en el capítulo siguiente que la solicitud de cada opción en este caso es muy parecida en cuanto a los datos solicitados para ser ejecutada.



1.3.5 Menú para programas (Program)

El menú Program manejará opciones que tengan que ver con los aplicaciones creadas posibles de ejecutar, tendremos:

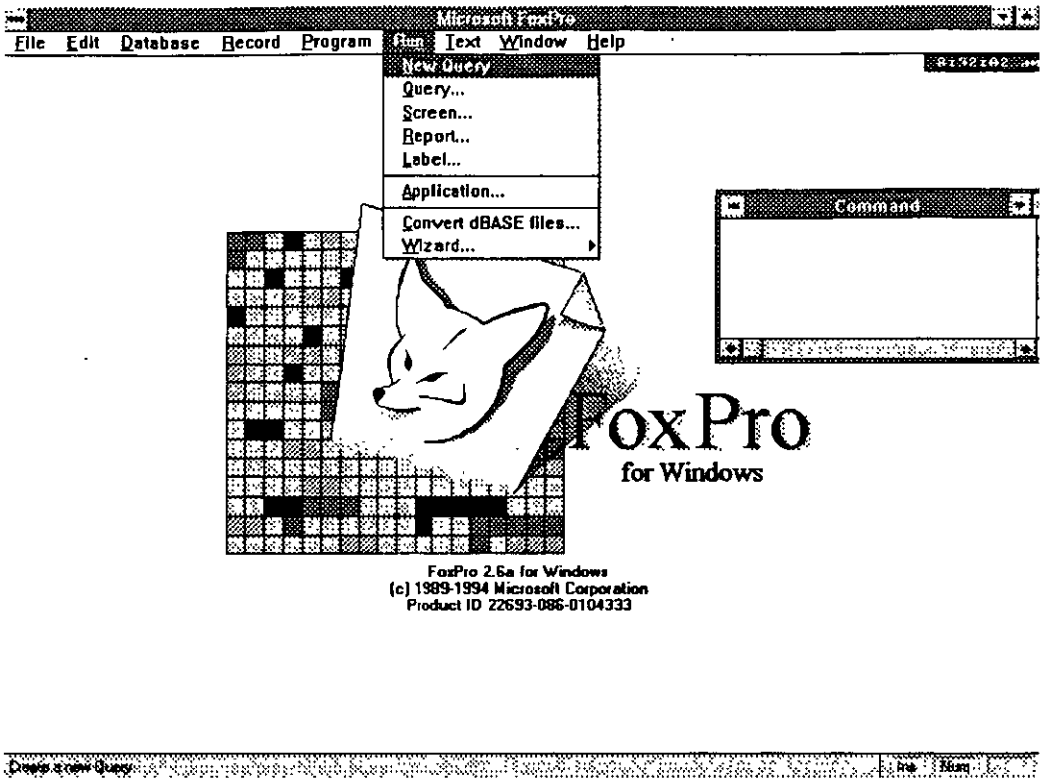
DO: Correr un programa desde el intérprete de Foxpro
Cancel: Cancela el programa
Resume: Suspende un programa.
Debbug: Llama el debugger de Foxpro, éste correrá interactivamente con el programa. Por medio de esta utilería es factible analizar el contenido de las variables que se encuentran vivas en ese momento en el programa que se encuentra corriendo.
Trace: Utilería que corre con el programa por medio del cuál se podrá ir manipulando la corrida de una aplicación, esto es, correr solo una subrutina, detener el programa para poder inspeccionar las variables por medio del debbuger. Esto nos hace notar que el debbuger y el trace van íntimamente relacionados. Ambas utilerías son muy útiles ya que para el caso de usuarios poco experimentados, es una forma de verificar el funcionamiento de los programas, así como encontrar errores de lógica difícilmente localizables
Compile: Para el caso de que no le establezcamos al compilador que en el momento de salvar un programa lo compile (Edit-Preferences-Compile when saved) y tomando en cuenta que cuando se corre lo compila automáticamente, es recomendable hacerlo para evitar problemas en cuanto a tratar con versiones compiladas anteriores a la modificada y para que en caso de existir algún error de sintaxis el sistema lo indique.
Generate: La opción de generación se utilizará en aplicaciones como los generadores de pantallas, menús, etc. Por medio de esta opción se generará código de la aplicación creada por medio de los menús para que pueda ser manipulada por el usuario. Esto resulta muy práctico para el usuario con cierta experiencia en programación.
Macros: Modificación y creación de teclas de macro instrucciones.
Beautity: Modifica los programas dándoles sangrías, encabezados, etc.
Foxdoc: Crea una documentación del sistema, tablas de referencia, indica las variables y bases utilizadas, etc. Veremos al final del curso estas dos utilerías con mayor detalle.



1.3.6 Menú para correr una aplicación

Menú donde se correrá alguna aplicación.

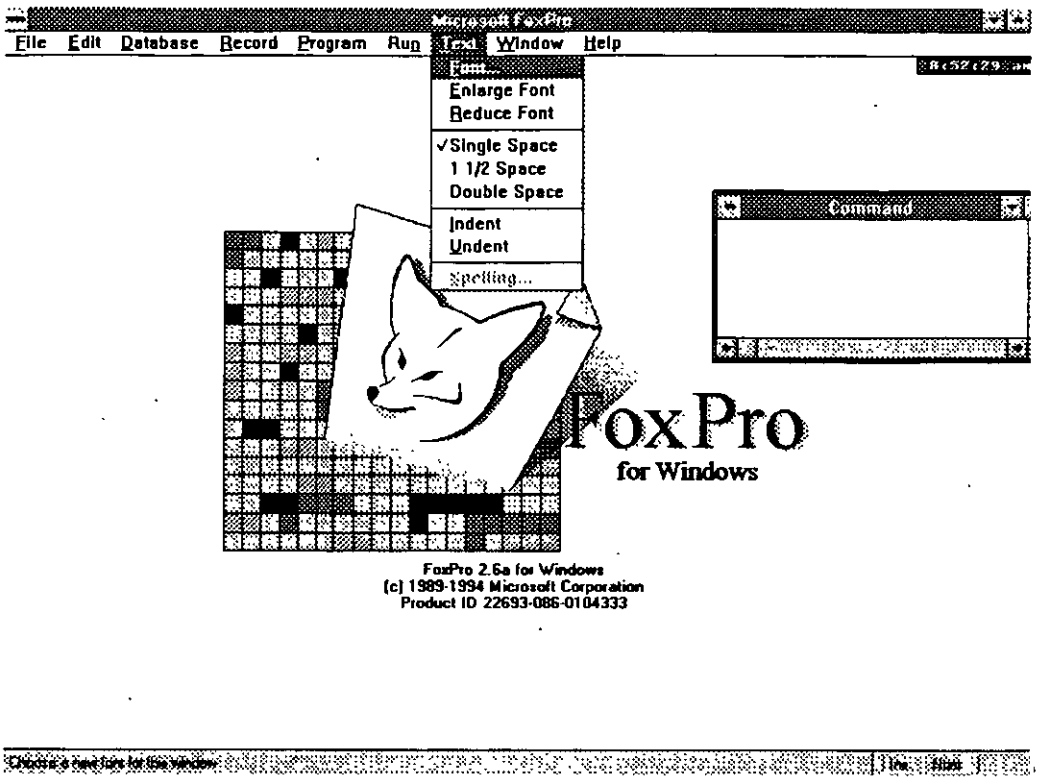
New Query: Crea un nuevo query.
Query: Corre un query elaborado previamente
Screen: Corre una pantalla diseñada por el usuario.
Report: Corre y despliega un reporte hecho.
Label: Corre y despliega etiquetas hechas con el reporteador.
Application: Corre un programa.
Convert dBase files: Convierte archivos, pantallas hechas en dDase III y IV a Foxpro.
Wizard: Corre el menú de wizards (Base de datos, query, pantallas, reportes, etiquetas, Mail merge)



1.3.7 Menú para el manejo de texto (Text)

Menú para el manejo de texto dentro de las aplicaciones y texto tecleado en el intérprete directamente, teniéndose las siguientes opciones:

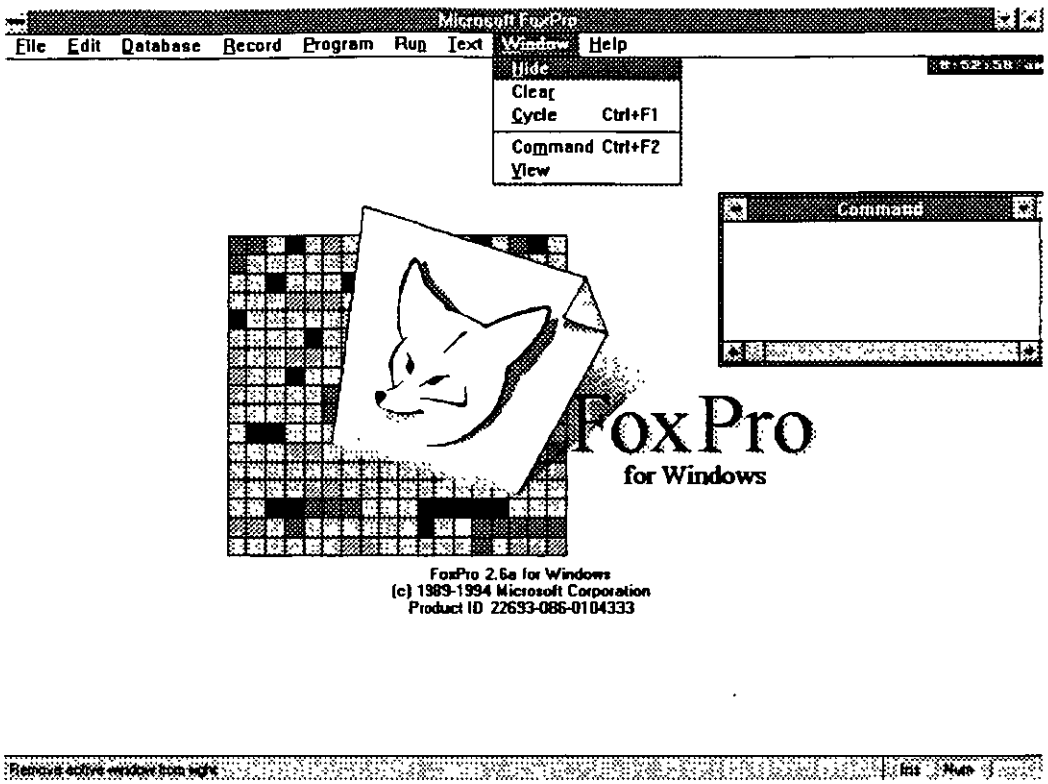
Font: Tipo de letra a utilizar.
Enlarge font: Agrandar la letra seleccionada.
Reduce font: Reducir la letra seleccionada.
Single space: Escribir a espacio sencillo.
1 ½ space: Escribir a espacio y medio.
Double space: Escribir a doble espacio.
Indent: Dar sangría
Undent: Quitar sangría.
Spelling: Revisar la ortografía.



1.3.8 Menú para la manipulación de ventanas (Window)

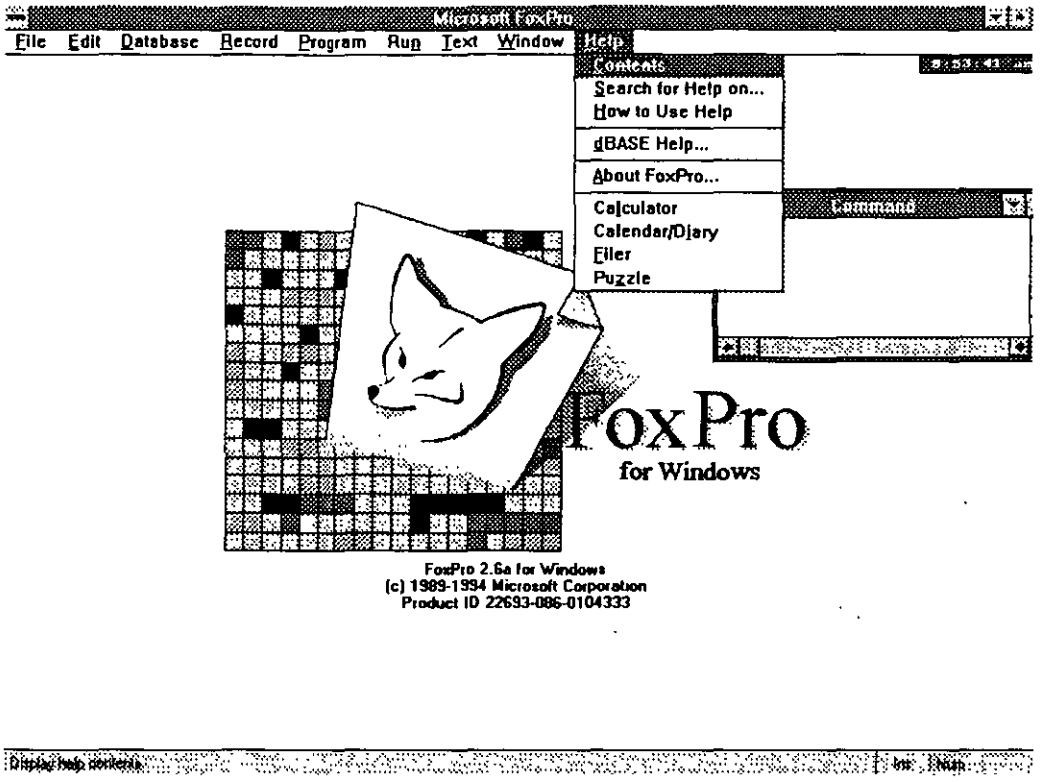
Este menú nos permite manipular ventanas, del sistema y ventanas creadas por el propio usuario.
Se tienen las siguientes opciones:

Hide: Esconde la ventana sobre la que se encuentra el cursor.
Clear: Limpia la ventana activa.
Cycle: Recircula las ventanas activas
Command: Envía el control a la ventana de Command
View: Muestra una ventana desde donde se pueden abrir bases de datos en diferentes áreas de trabajo, si existen abiertas nos las mostrará.



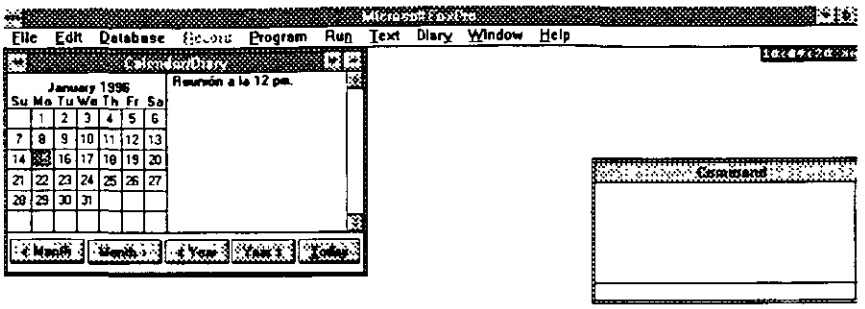
1.3.9 Menú de ayuda y miscelánea (Help)

Aquí encontraremos las diferentes llamadas a las ayudas que nos proporciona el paquete, así como otro tipo de aplicaciones que el usuario encontrará de gran utilidad para sus tareas, trabajando desde el ambiente.

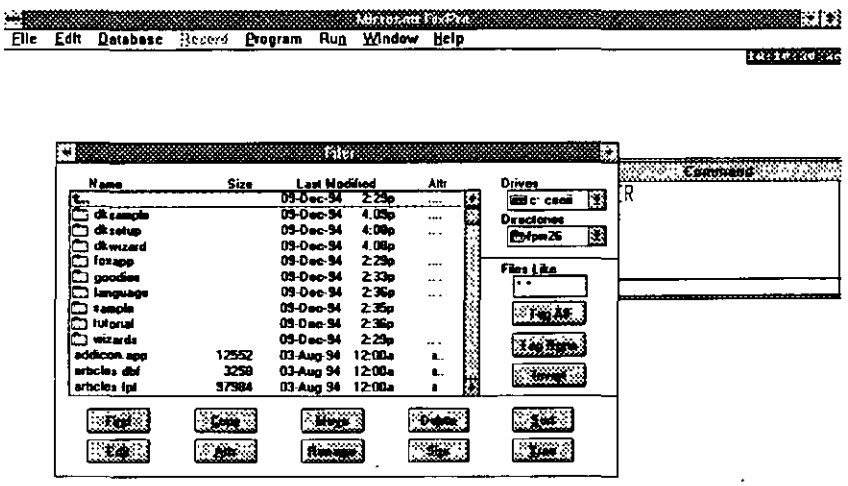


Se presentan las siguientes opciones:

Contents: Muestra el contenido de la ayuda.
Search for help on: Busca algún tópico específico.
How to use help: Texto que nos da una orientación de como utilizar la ayuda.
dBase help: Activa la ayuda de dBase
About Foxpro: Referencias del paquete.
Calculator: Calculadora.
Calendar/diary: Calendario diario donde nos permitirá tener un calendario mensual e ir anotando día a día algún asunto a tratar, de tal forma que al consultarlo nos despliegue la nota del día.



Filer: Es una especie de administrador de archivos. Donde vamos a poder borrar, editar, copiar, mover, renombrar archivos en todas las unidades de disco y todos sus subdirectorios.
Es una aplicación muy utilizada ya que permite realizar transacciones con archivos sin salir del intérprete, ni hacer un shell a DOS lo que a veces resulta un poco tardado.



Puzzle: Juego de ordenación de números de Foxpro.

2. Creación y manipulación de bases de datos

El FoxPro se fundamenta en el lenguaje X-Base estándar con el que tiene algunas diferencias notables.

2.1 Definición de una base de datos

Una base de datos comprende un conjunto de información ordenada según un formato determinado, que contiene como unidad esencial el registro, el cual se divide a su vez en campos. Un campo está compuesto de información que es la parte mas pequeña de una base de datos.
Cada base de datos contiene numerosos registros que pueden estar relacionados con diferentes bases de datos
Un modelo *relacional* es aquel que tiene acceso a la información cualquiera que sea la dirección en la que se intente coordinar los datos entre sí.
Fox pro permite utilizar un total de 99 archivos de base de datos, de los cuales un máximo de 25 pueden permanecer abiertos al mismo tiempo.

2.2 Datos que componen una base de datos

Nombre de campo (Name): Nombre que llevará el campo, su longitud máxima será de 10 caracteres y no se admiten numéricos como primer carácter, es posible utilizar como símbolo de unión el subguión (_) y el guión (-)

Tipo del campo (Type): La naturaleza del campo. Como tipos válidos de campo en Foxpro tenemos los siguientes:

Tipo de dato	Características	Almacén
Caracter (Character)	Alacena números, letras y símbolos	de 1 a 254 por campo

Tipo de dato	Características	Almacén
Numérico (Numeric)	Almacena números y su símbolo con precisión sencilla	de 1 a 26 incluyendo signo y punto decimal
Flotante (Float)	Almacena números manejando doble precisión	de 1 a 26 incluyendo signo y punto decimal
Fecha (Date)	Almacena fechas en diferentes formatos	8 espacios
Lógico (Logical)	Almacena solo valores verdadero y falso (T o F)	1 espacio
Memoria (Memo)	Almacena texto teniendo como máximo la capacidad de disco de la máquina	10 espacios en su descripción
General	Alacena imágenes y sonido	10 espacios como definición

Tamaño (Widht): Longitud del campo
Decimales (Dec): Número de decimales

2.3 Creación de una base de datos

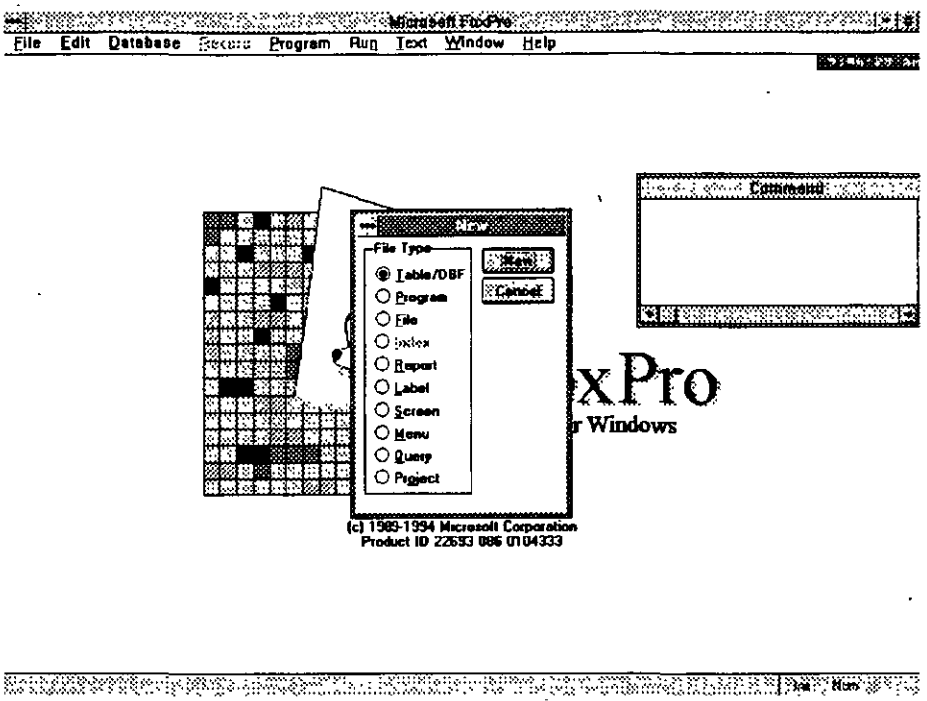
Para crear una base de datos veremos diferentes métodos, el primero es por medio del menú, el segundo es tecleando desde la ventana de command el siguiente comando:

CREATE <nombre>

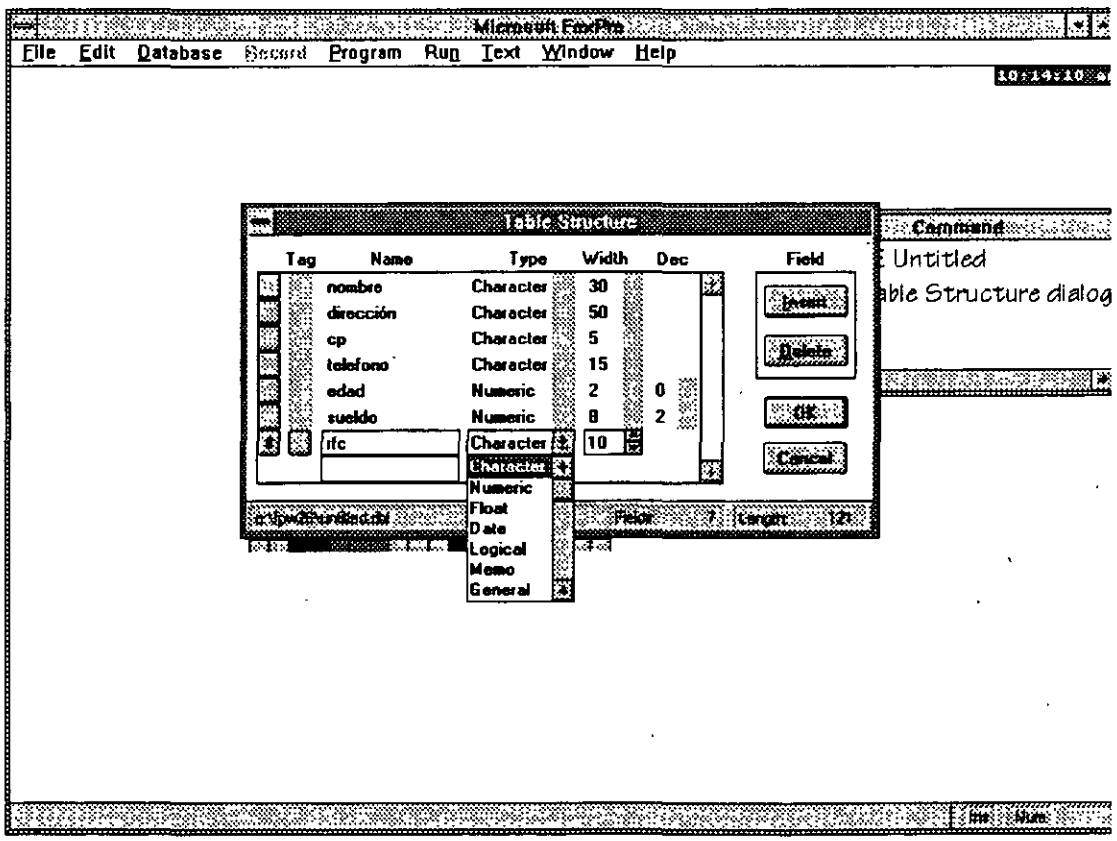
y nos llamará directamente a la pantalla de dar de alta la base de datos.

A continuación detallaremos el proceso entrando desde menú.

Como recordaremos para crear un archivo nuevo cualquiera que sea su naturaleza llamaremos al menú File-New y tendremos la siguiente pantalla donde nos solicitará el tipo de archivo a crear, en este caso será Table/DBF (base de datos).



Una vez seleccionado tendremos una nueva pantalla donde definiremos los campos que compondrán a la base de datos.

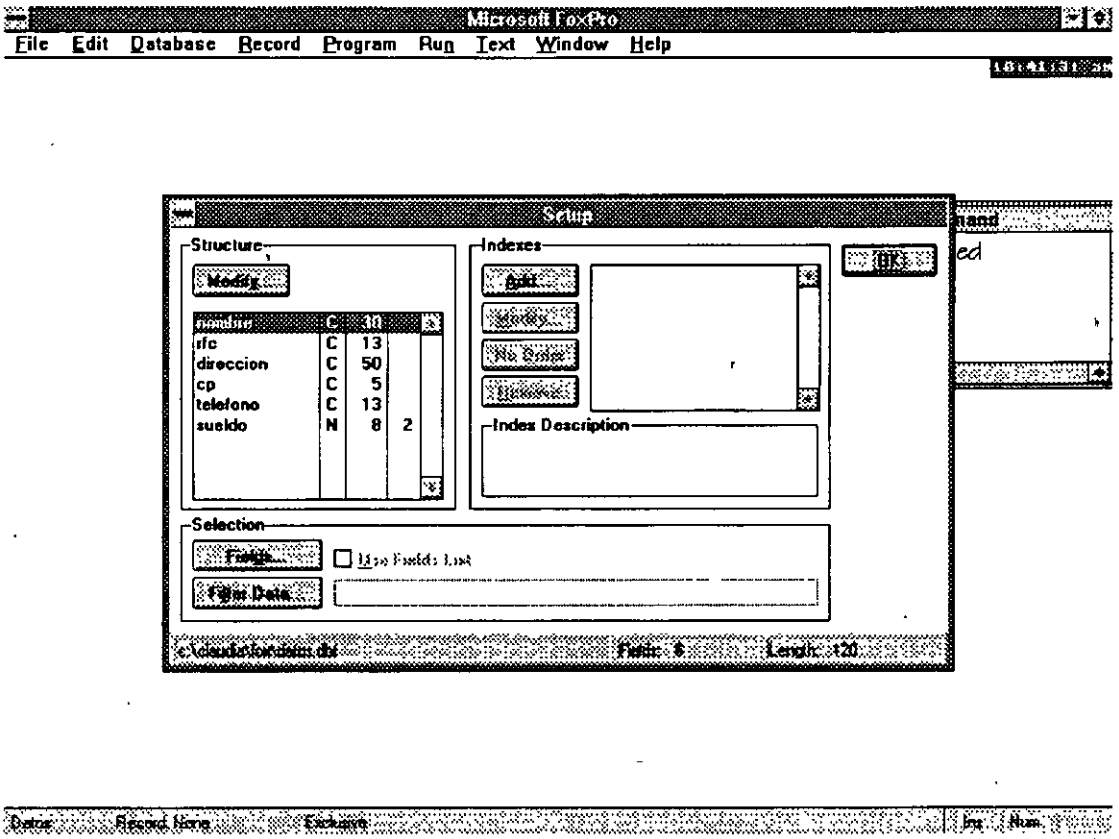


Ventana de creación de base de datos

En la pantalla anterior notamos que existe dos columnas anteriores a lo que identificamos como el nombre del campo.
Para el caso de la barra mas a la izquierda nos servirá para desplazar los campos de lugar, esto es, pensemos que una vez tecleado no nos gustó el orden y desearíamos que el rfc estuviera después del nombre, bastaría con seleccionar con el mouse el botón izquierdo de este campo y arrastrarlo al lugar deseado.
Por lo que se refiere a la segunda columna la utilizaremos para definir los índices. Esto se verá en el siguiente capítulo.
Después escribiremos el nombre del campo, nos desplazamos al siguiente lugar por medio del tabulador y seleccionaremos en tipo de datos a asignarle a ese campo. Notemos que se nos presenta una lista desplegable con las opciones válidas de las cual seleccionar.
El siguiente es la longitud del campo y por último el número de decimales. Para el caso de campos numéricos con decimales deberán de tomarse en cuenta como longitud el punto y los decimales. Por ejemplo si necesitamos como máximo almacenar un número 999,999.99 tendremos como longitud 9 y dos decimales, que serán 6 enteros+el punto+dos decimales=9.
Para el caso de haber olvidado algún campo a intercalar entre otros ya tecleados, bastará con colocar el mouse donde se desea agregar el campo y oprimir el botón de insertar colocado a la derecha de la pantalla, lo mismo sucede si se desea borrar un campo.
Cuando los datos están correctos se oprimirá el botón de Ok. El sistema preguntará el nombre que le daremos a la base de datos y si se desea en ese momento teclear datos en la base de datos nueva.
Si por error antes de terminar de teclear todos los campos se oprima <enter> nos mandará a una pantalla de salvar el archivo preguntándonos el nombre, opriman <esc> para el caso de querer continuar. En caso contrario teclee el nombre que tendrá la base y en seguida el compilador preguntará si se desea dar entrada a datos en este momento.

2.4 Modificación de una base de datos

Para el caso de que la base de datos creada una vez que se ha tecleado y salvado se necesita modificar se deberán de realizar los siguientes pasos:
Si la base no se encuentra abierta, abrirla. (File-Open)
Irnos al menú de modificación (Database-Setup), que nos presentará lo siguiente:



La pantalla mostrada está compuesta de varias partes, por el momento sólo atenderemos a la parte izquierda superior de la pantalla. Aquí se nos muestra una tabla con la definición de la base de datos y en la parte superior un botón que nos permitirá modificar (Modify).
Si oprimimos el botón de modificar tendremos nuevamente la ventana de la estructura de la base de datos vista anteriormente, en la cuál se nos permitirá modificar nombres de campos, longitudes, agregar campos, borrar campos, modificar el orden.

Para el caso de modificar el tipo de campo se deberá de tener precaución porque es posible llegar a perder información, por ello se recomienda no hacerlo antes de haber obtenido una copia de la base de datos.

2.5 Manipulación de la información

Veremos los comandos para manipular la información existente en una base de datos.

2.5.1. Abrir (Use)

Para abrir una base de datos existente desde menú utilizaremos el menú File-Open, el sistema nos preguntará el nombre de la base a abrir. (referencia el primer capítulo).
Si se desea utilizar la instrucción, bastará con teclear en la ventana de command lo siguiente

USE <nombre de la base de datos>

Por ejemplo:

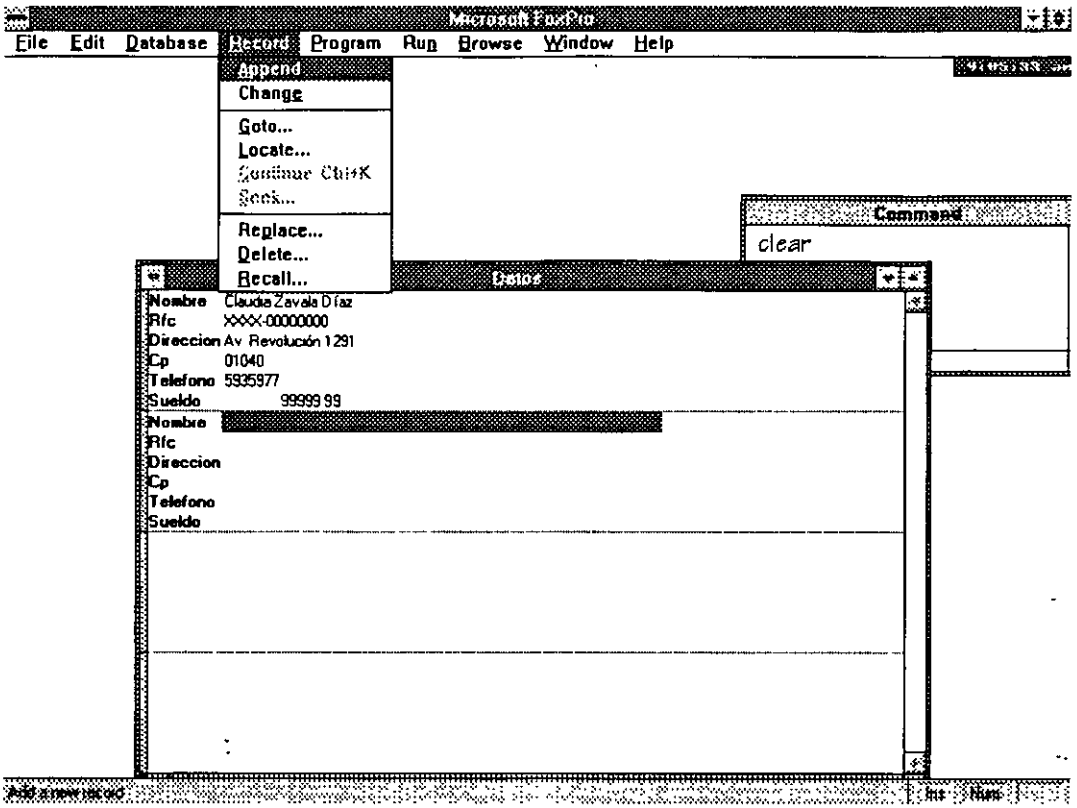
USE DATOS_G

Automáticamente desplegará en la barra de estado el nombre de la base y el número de registros que posee.

2.5.2. Agregar (Append)

El comando append o agregar se utilizará para anexar información a la base de datos ya creada y activa en ese momento. El sistema nos mostrará una ventana en la cuál aparecerá la estructura de la base de datos y el cursor a partir del cuál podemos iniciar a teclear información. Esto será registro por registro.

Por medio del menú podremos anexar los datos, utilizaremos el menú Record-Append y el sistema nos mostrará lo siguiente:



Notamos que en la pantalla aparecen los nombres de los campos y nos desplegará espacio para registros nuevos en el momento de terminar de capturar uno.
Una vez terminada la captura podremos salir de la pantalla con Ctrl-W para salvar los cambios.

Para el caso de querer utilizar la instrucción directamente, desde la ventana de command teclearemos lo siguiente:

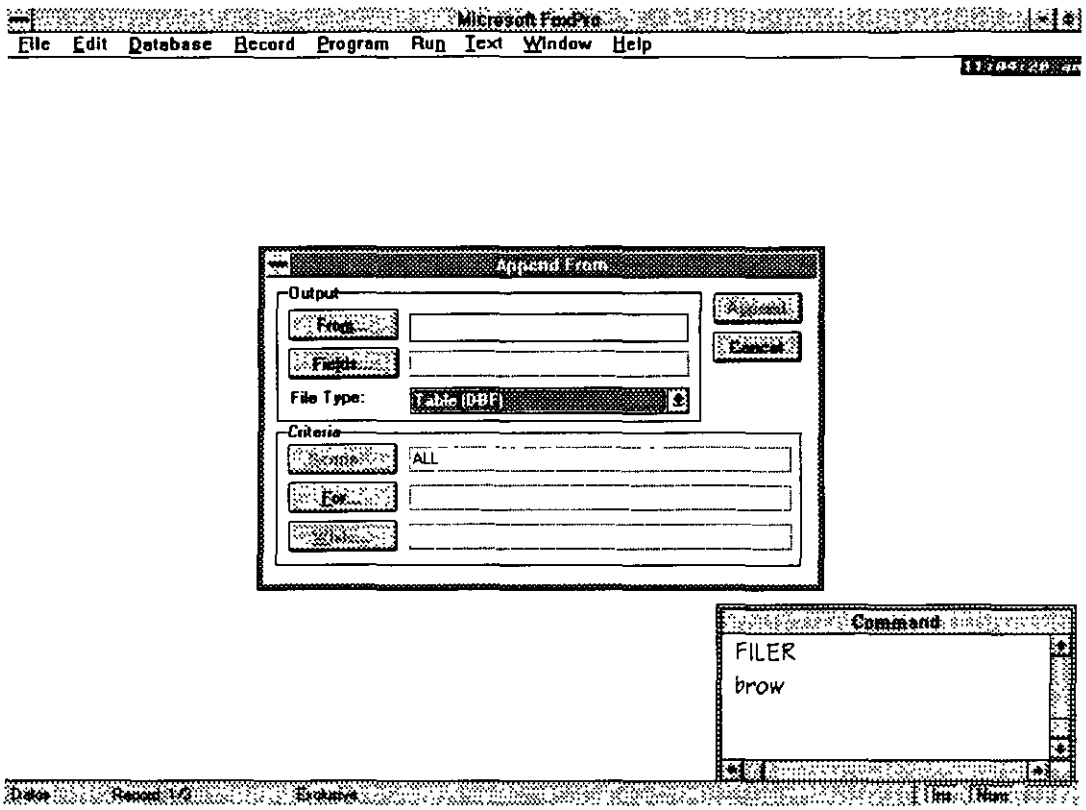
APPEND

En ese momento se nos mostrará la pantalla desplegada anteriormente.

También es posible agregar información de una base de datos existente a la activa, esto lo haríamos mediante el comando:

APPEND FROM <BASE ORIGEN>

Es posible utilizar también el menú llamando a Database-Append From presentándonos lo siguiente:



En esta pantalla se nos preguntará:
From: Base de datos o archivo de donde se obtendrá la información a agregar.
Fields: En caso de que se desee especificar algún o algunos campo(s) en especial, de lo contrario se deja en blanco y tomará todos.
Field Type: Tipo de archivo de donde tomará la información. Aquí se le definirá si es una base de datos o archivo y su formato.
Scope: Cuantos registros se agregarán.
For: Con que condición. Por ejemplo : Todos los que su RFC inicia con A
While : Condición que agregará mientras una condición se cumpla. Por ejemplo: Mientras el sueldo sea mayor a 10,000.

Cabe aclarar que en el caso de agregar información de base a base sólo se agregarán aquellos campos en que los nombres origen sean iguales.

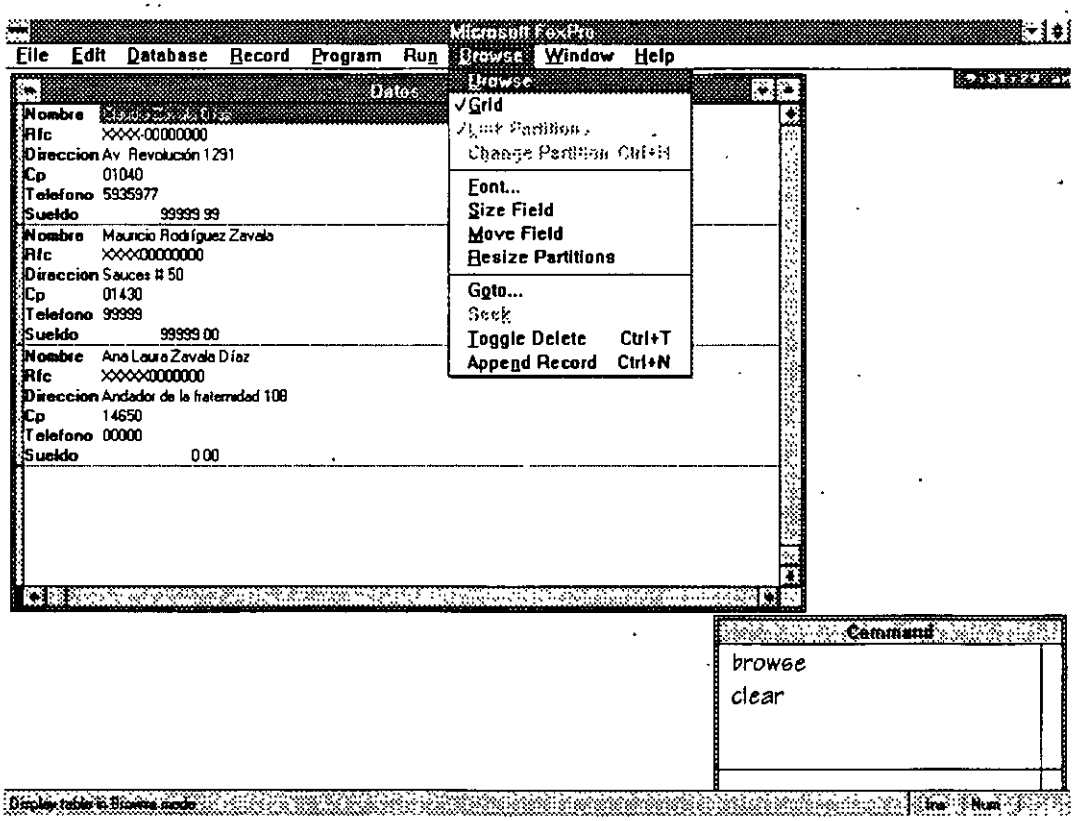
2.5.3. Modificar (Browse)

El comando Browse nos permitirá desplegar la información existente en nuestra base de datos y poder así modificar cualquier dato en ella. Por menú llamaríamos a Database-Browse mostrándonos el contenido de la base de datos.

Para el caso de querer utilizar la instrucción teclearemos en la ventana de command lo siguiente:

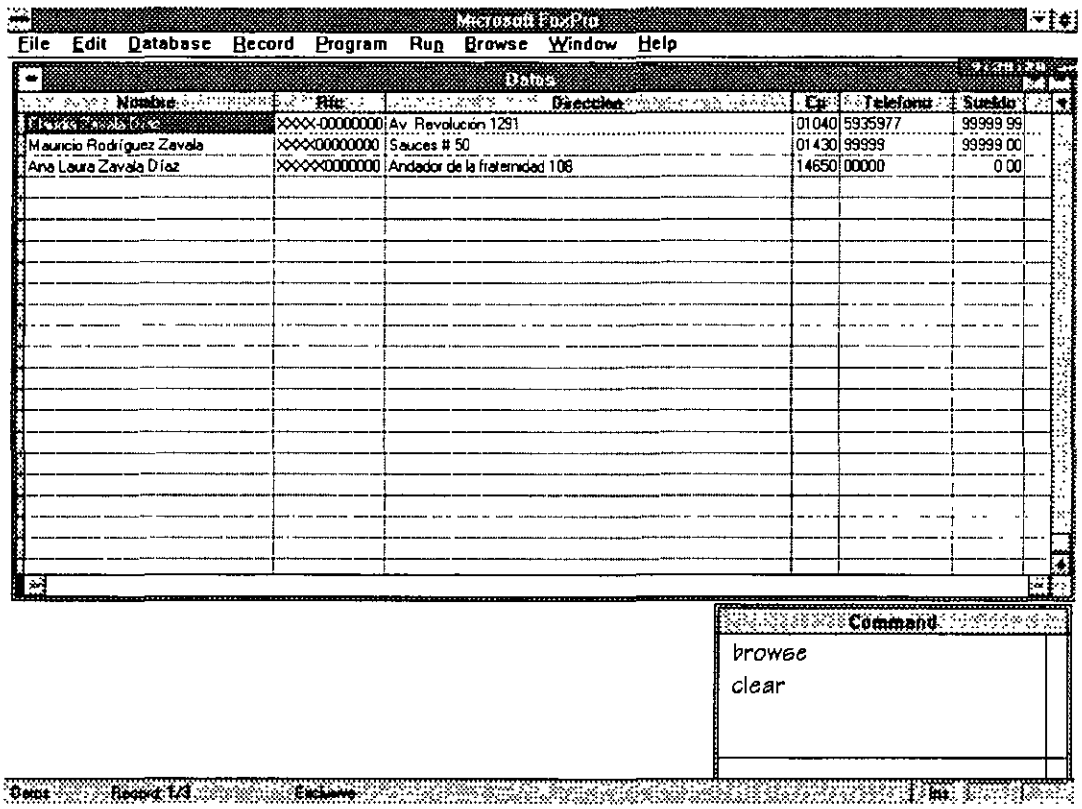
BROWSE

En ambos caso nos desplegará la misma pantalla, en este momento del despliegue si hechamos un vistazo al menú principal notaremos que ha sido modificado apareciendo una nueva opción denominada Browse.



La nueva opción en el menú llamada Browse nos presenta varias alternativas:

Browse/Change: Esta opción nos permitirá poseer diferentes vistas de la misma base de datos, la primera es la mostrada anteriormente, en esta el despliegue de la información se hace por medio de registros en donde tendremos en forma de renglón cada campo de la base de datos y separados por líneas un registro de otro. La segunda forma de despliegue nos presentará la pantalla siguiente:



En el momento de cambiar el tipo de despliegue cambiará la palabra Browse dentro de este mismo menú por la palabra Change.

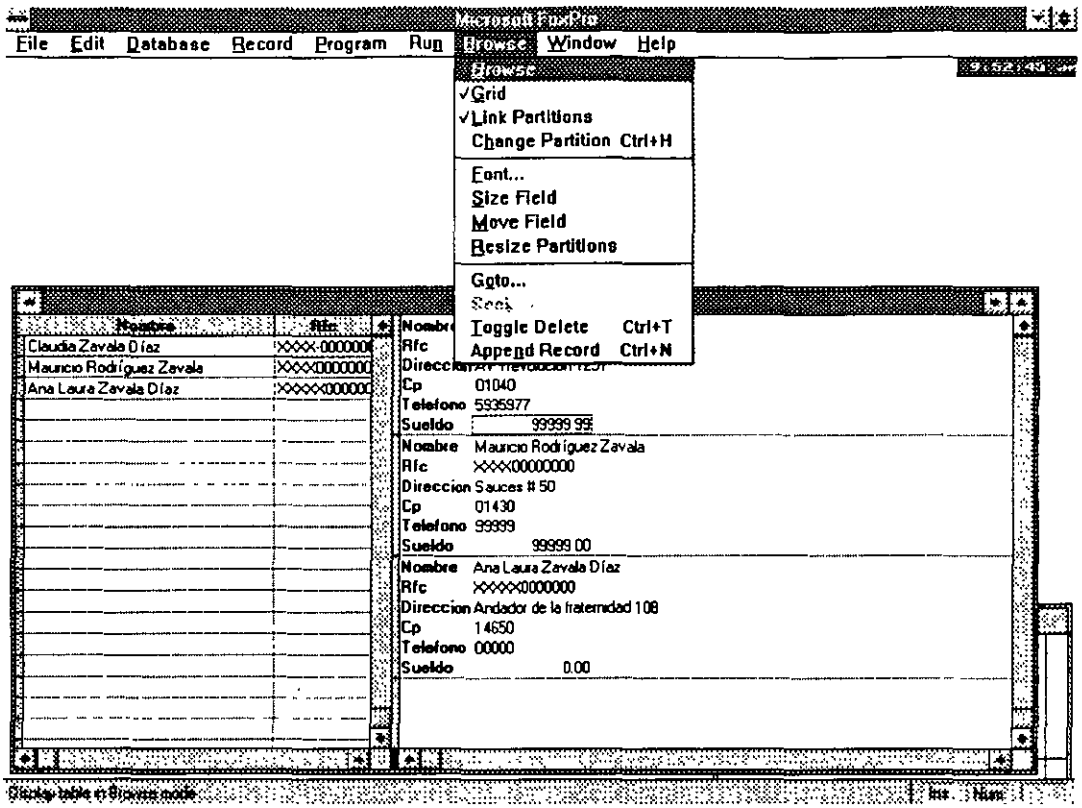
En la pantalla anterior se nos muestra cada registro como un renglón y los campos como columnas y no los campos como renglones, como lo es el caso de Browse.

Para modificar al tipo de despliegue anterior bastará con seleccionar Change del menú Browse.

Grid: Eliminará o incluirá las rayas de división entre registros y campos.

Link partitions: Liga o desliga las particiones hechas

Change partition: Cambia el cursor de partición en partición.
Font: Modifica el tipo de letra utilizado en la base de datos en el momento del despliegue.
Size: Cambia el tamaño de un campo, sólo en forma visual y durante el despliegue. Esto se hará de la siguiente forma, pensemos que tenemos un campo que es demasiado grande y no permite que pueda ver en la pantalla otros campos y no es necesario que esté desplegado todo, por lo que desearía hacerlo mas pequeño sin modificar la estructura de la base se datos. Se coloca el cursor en el campo seleccionado y llamamos al menú Browse seleccionando la opción size, ahora me muevo con las flechas de dirección para la derecha si lo deseo hacer mas grande o la de la izquierda si lo quiero reducir. Otra forma es sin llamar al menú colocarse el renglón donde se encuentran los nombres de los campos sobre la línea divisoria entre campo y campo y aparecerá una cruz indicándonos que es posible desplazarla, de esta forma podremos obtener el tamaño que se desee. (esta última forma es muy utilizada en aplicaciones en windows como Excel y Winword).
Move Field: Mueve un campo de lugar. De la misma forma pensemos que durante el despliegue me gustaría que apareciera el sueldo después del rfc solo para consulta. Nos colocaremos en el campo que deseamos desplazar y seleccionaremos del menú de Browse - Move field e iniciaremos a desplazarnos arrastrando el mouse o desplazándonos con las flechas.
Resize partitions: Llamaremos particiones al poder tener dos vistas de la misma base de datos al mismo tiempo en la pantalla. Por lo que los comandos anteriores de Link partitions (Ligar particiones) y Change partitions solo se activarán si existe algún tipo de partición.
Para crear las particiones llamaremos al menú Browse - Resize partitions y aparecerá en la pantalla dos líneas verticales con una a la flecha izquierda y otra a la derecha, indicándonos desplazamiento (<-II->). Con las flechas de desplazamiento nos moveremos hacia la derecha, apareciendo dos líneas verticales que abarcan a todo el despliegue de la base de datos. Nos moveremos hasta el tamaño deseado y <Enter> en ese momento tendremos las pantalla dividida en dos con dos vistas distintas de la base de datos, como se muestra en la pantalla siguiente:



Esto resulta muy útil cuando se desean consultar diferentes datos de una misma base.

Goto: Ir a algún registro en específico.

Seek: Busca un dato en especial, sólo en bases de tipo indexado.

Toggle delete: Marca un registro para ser borrado. Cuando un registro ha sido marcado para ser borrado aparecerá en la columna más a la izquierda que se tiene en el despliegue de la base, una marca negra indicando que se borrará.

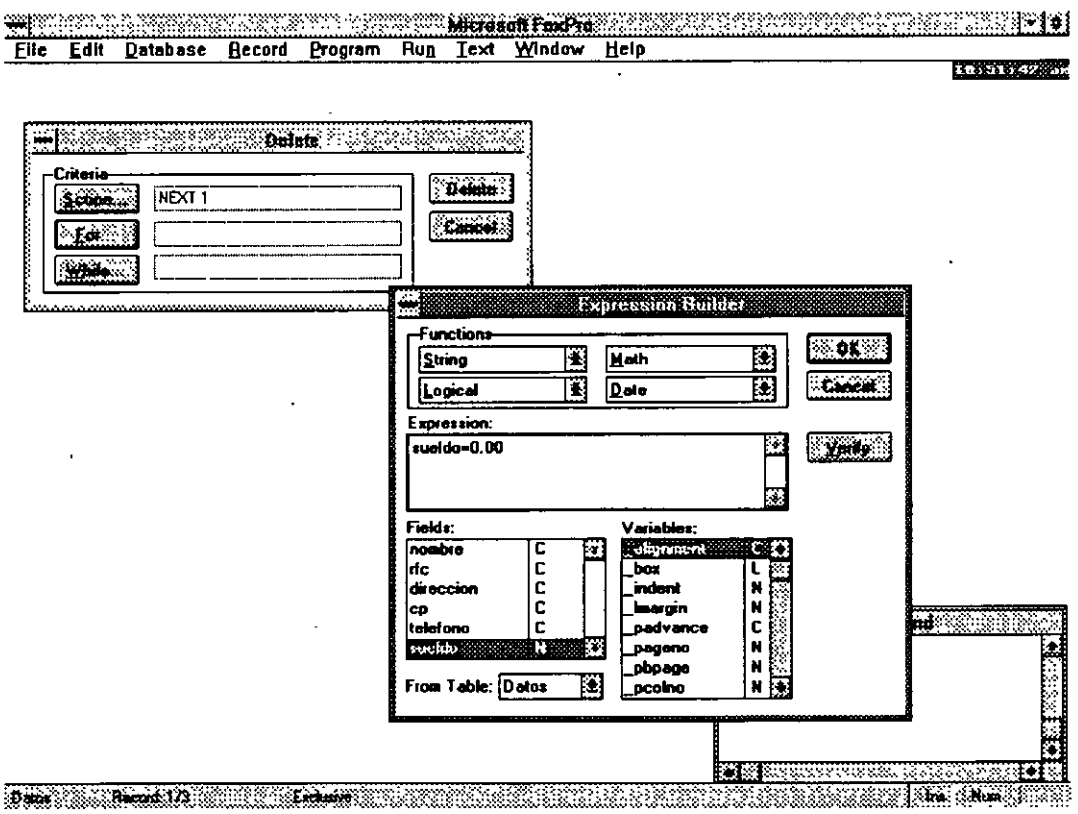
Para poner y quitar esta marca sin la ayuda del menú bastará seleccionar o deseleccionar esta área con el mouse.

Append record: Agregar un registro nuevo estando dentro del despliegue, ya que si se seleccionó la opción browse, por naturaleza no nos permitirá agregar información.

2.5.4 Borrar (Delete)

Para borrar información tendremos varios comandos y formas de hacerlo.
Registros específicos sin ninguna relación: Como se describió anteriormente marcándolos en la columna izquierda durante el despliegue.

Registros con alguna condición: Es posible borrar sólo determinados registros. Por ejemplo todos los que en sueldo tienen 0.00. Para ello tendremos dos métodos por medio de menú y por comando.
Por menú llamaremos a Record-Delete y se nos mostrará la siguiente pantalla:



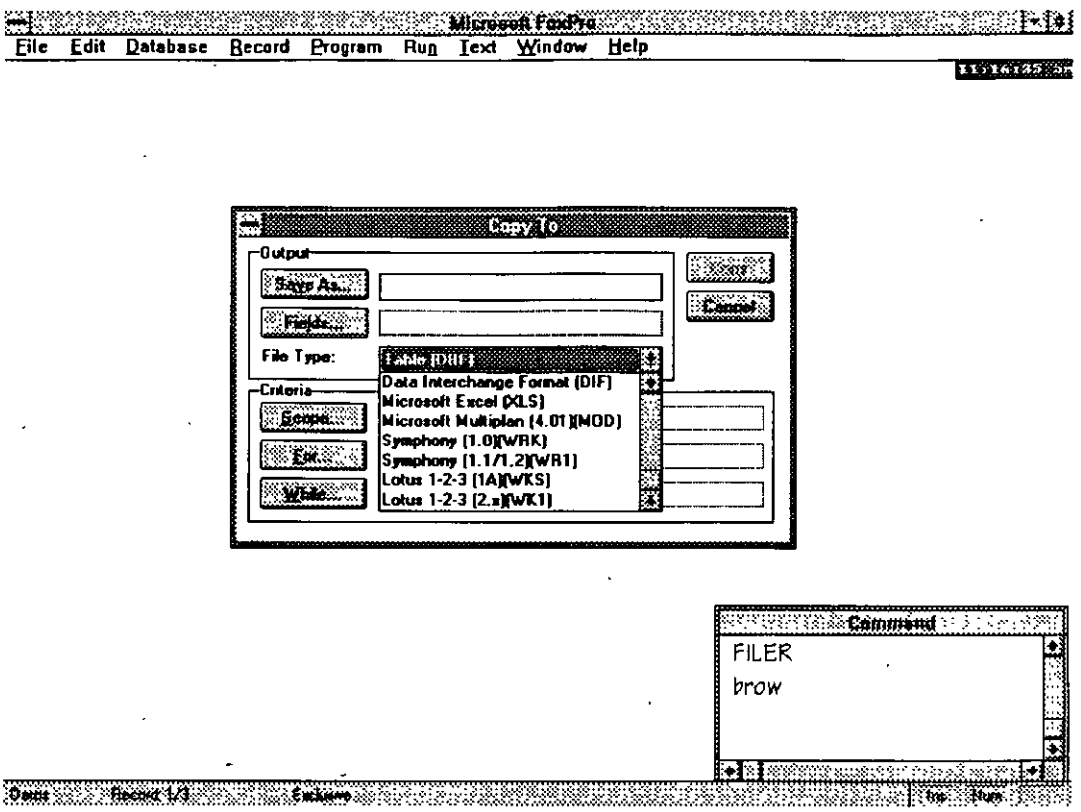
Aquí se nos preguntará lo siguiente:

Scope: Es el número de elementos a tomar en cuenta. Por default nos dice que el primero ALL tomaría a todos.
For: Podremos determinarle una condición. Una vez que se oprimió el botón For aparece una nueva pantalla donde nos muestra los campos y diferentes funciones a utilizar, tecleando en *Expression* la expresión de condición, por ejemplo de todos aquellos que tienen en sueldo 0.0, como se muestra en la pantalla. Esto hará que se marquen para borrar todos los registros que caigan en este caso.
While: El manejo es el mismo sólo que en este caso el concepto es borrar mientras una condición se cumpla. Por ejemplo: borra mientras el sueldo sea menor o igual a cero, esto es, borarrá hasta que se encuentre con el primer registro que no cumpla con la condición dada.

Toda la base: Delete all o Zap en ambos casos eliminará toda la información, la diferencia radica en que con Delete es posible recuperarla porque sólo la marca y con Zap en el momento de aceptar este comando eliminará toda la información.

2.5.5 Copiar (Copy)

El comando copiar se utilizará para copiar la base de datos abierta a un archivo con un formato específico. Por ejemplo: a una hoja de cálculo de Excel, o a un archivo ascii separando los campos por blancos, etc.
Mediante el menú tendríamos una llamada al menú Database-Copy to y se nos mostrará la siguiente pantalla:



Donde le determinaremos el nombre del nuevo archivo, como en otros comandos los campos a copiar, las condiciones y número de elementos, pero tendremos una amplia lista de formatos diferentes para cambiar a nuestra base de datos y poderla exportar a otra aplicación.

Para el caso de hacerlo mediante la ventana de command tendríamos que teclear lo siguiente:

COPY TO <NUEVO NOMBRE> <FORMATO>

Por ejemplo para exportar a una hoja en Lotus se tendría

COPY TO HOJA_123 WK1

Los tipo de formatos los encontraremos en la lista que se nos mostró anteriormente en la ventana de despliegue, este aparecerá al lado de la descripción de la aplicación a la que deseamos enviar la base encerrado entre paréntesis.

3. Índices

3.1 Definición

Para que una búsqueda sea más rápida, una base de datos debe clasificarse ordenándola por su campo más significativo, como podría ser la clave del empleado, el código, etc.
Un archivo índice consta de por lo menos un campo de la base de datos. Este campo es posible ordenarlo alfabéticamente, numéricamente o bien cronológicamente, unido a él nos encontraremos con el número de registro que se utiliza como referencia en la base de datos primaria u original. Un archivo de índice es una ordenación virtual de la base de datos original.
De la misma manera que el índice de un libro es una sección independiente que indica dónde se puede localizar la información, los archivos de índice de FoxPro constituyen archivos independientes que contienen información relativa a la localización de registros individuales en la base de datos original. Cuando se abre un archivo de base de datos junto con el archivo de índice, el primer registro que se recupera no es el que ocupa la posición inicial en la base de datos de partida, sino el primero que figura según el tipo de índice que se generó (depende del tipo de ordenación dada). Es importante mencionar que la indexación no afectará el orden natural de los registros, entendiendo como orden natural el que va teniendo la información dependiendo de su captura.

Aunado a la definición de índice definiremos la palabra LLAVE, que conoceremos por el dato por medio del cuál se indexará. La idea de una llave es tener una clave única por la cuál identificar a sólo un registro en especial. Cuando manejamos llaves con mas de un dato o con campos que pudieran estar repetidos como: sueldo, número de depto, etc. básicamente se utilizará para ordenar la información bajo algún criterio.

3.2 Tipos de índices

Todas las versiones de FoxPro a partir de la 2 constan de dos tipos de índices, cualquiera de ambos métodos nos llevarán al mismo resultado de organización de la información.
Tendremos índices CDX e IDX, los cuáles veremos con mas detalle a continuación.

3.2.1 Índices CDX

El primer método consiste en índices compuestos que poseen la extensión .CDX. Este archivo contiene información sobre los diferentes índices que hayan sido creados a partir del archivo original. Este tipo de índices puede contener diferentes tipos de ordenación de la información, pero sólo una de ellas estará activa.
Este tipo de índices llamados .CDX estructurales se abren y actualizan de forma automática cada vez que se abre la base de datos, por lo que no será necesario abrirlos, pero si indicarle cuál será la ordenación activa.
El archivo CDX se llamará por default de la misma forma que la base de datos original.
Para poder identificar las diferentes formas de ordenar una base de datos teniendo sólo un archivo índice CDX, seleccionaremos TAGs o llaves distintas por medio de las cuáles se hará referencia a cada una de organizaciones de la información.

Por ejemplo tenemos la siguiente base de datos

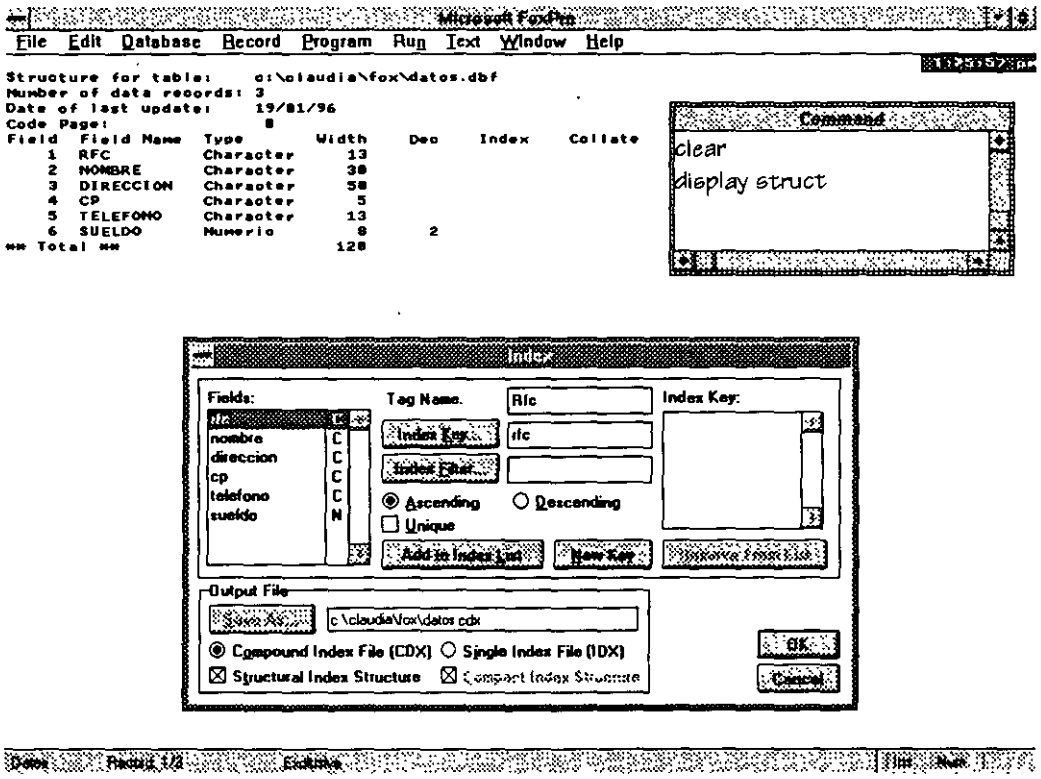
DATOS.DBF

RFC	C	13
NOMBRE	C	30
DIRECCION	C	50
CP	C	5
TELEFONO	C	13
SUELDO	N	8

y deseamos tener una ordenación por RFC y otra por NOMBRE, entonces crearemos 2 diferentes TAGs o etiquetas de índice. Un tag rfc y un tag nombre bajo del nombre del índice DATOS.CDX

3.2.1.1 Creación de un índice CDX

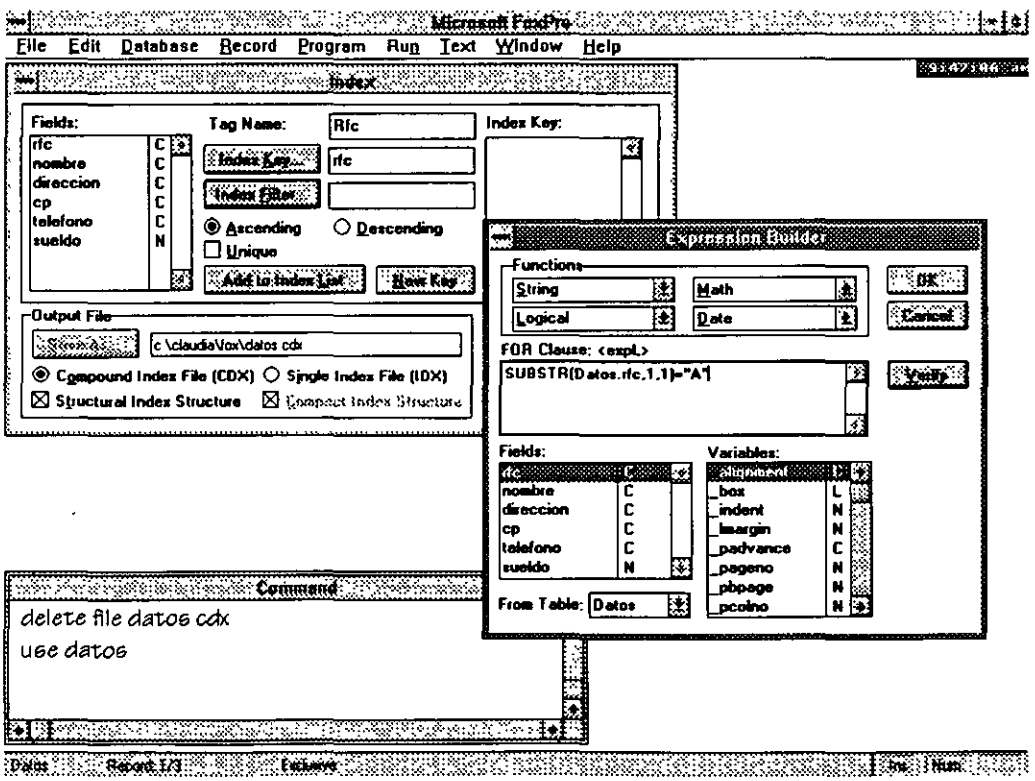
Analizaremos un archivo de índices CDX. Para crear un índice por menú llamaremos a la opción File-New-Index y nos aparecerá la siguiente pantalla donde identificaremos distintas partes, haremos referencia a ellas por medio de los encabezados que poseen:



Fields: Son los nombres de los campos que posee la base de datos, así como su tipo, esto nos ayudará a no mezclar tipos diferentes de datos sin transformar para conformar un mismo índice.
Tag name: Nombre del tag, de preferencia se le debe de llamar como el campo al que hace referencia para indexar, más no es indispensable.
Index key: Nombre del campo por el que se indexará.

Output File: Por default veremos que posee el mismo nombre que la base de datos y nos marca el tipo de índice de que se trata, en este caso Compound Index File (CDX).
En este caso tendremos opciones a elegir, CDX estructural o no estructural.
Para el caso de estructural sólo podremos llamar una sola vez a cada campo para formar los distintos tags del índice CDX, esto lo notaremos ya que en este caso una vez seleccionado un campo para indexar sobre de él, quedará desactivado para volverlo a usar.
Cuando seleccionamos no estructural, el sistema nos dará la opción de seleccionar las veces que sean necesarias los campos que se deseen.
Ascending/descending: Es posible que la ordenación sea ascendente o descendente.
Index Filter: Filtro para la creación de índices, esto es podríamos tener un índice que solo considerara a las personas cuyo RFC inicia con A.

Ilustrando el ejemplo anterior tendríamos



Podemos ver que cuando es utilizado un filtro llama a una pantalla anexa donde le determinaremos la construcción de este.
Después de crear el índice que se mostró en la pantalla anterior si creamos una vista de la base de datos, solo veremos a aquellos datos que cumplan con la condición determinada.

Para crear índices del tipo CDX desde la ventana de command tecleareemos lo siguiente:

INDEX ON <expresión> TAG <nombre etiqueta> FOR <condición>

Por ejemplo:

INDEX ON RFC TAG RFC FOR SUBSTR(RFC,1,1)="A"

3.2.1.2 Manipulación de los diferentes TAGS

Como se dijo anteriormente en el momento de abrir una base de datos que contiene un archivo índice CDX, éste se abrirá al mismo tiempo, pero ninguno de los TAGS estarán en ese momento activos, por lo que si se visualiza el contenido de la base la veremos en su forma natural.
Durante la programación para poder alternar entre los diferentes órdenes o TAGS del mismo índice CDX se deberá de tener en cuenta que será necesario abrir la base en forma indexada y utilizaremos la siguiente instrucción:

SET ORDER TO TAG <nombre del TAG>

Por ejemplo:

SET ORDER TO TAG RFC o SET ORDER TO TAG NOMBRE

De esta manera podemos alternar entre las diferentes formas de órdenes dentro de un mismo índice.

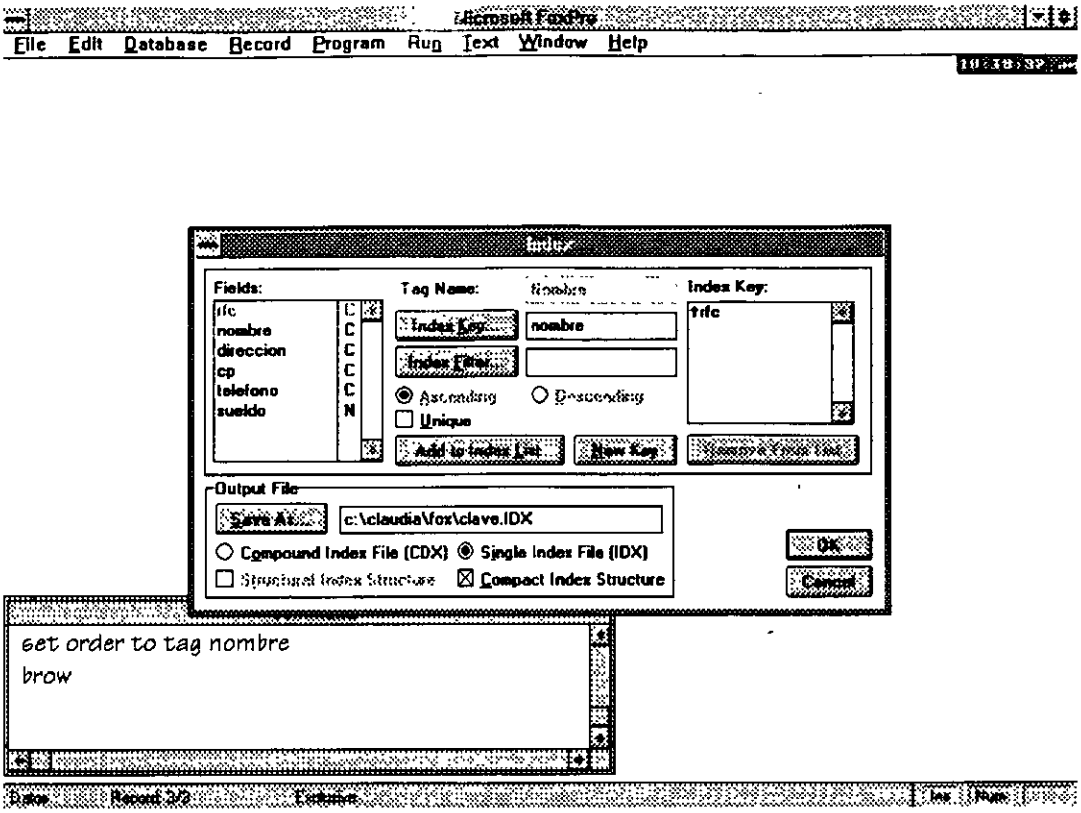
3.2.2 Índices IDX

Por lo que se refiere a los índices IDX o llamados índice individuales. Con este procedimiento, la información correspondiente a cada índice se almacena en un archivo independiente. Si fuera necesario indexar por el nombre y el rfc, se tendría que crear dos índices distintos. Estos archivos IDX pueden ser de dos tipos compactos y no compactos. Los primeros requieren menos espacio en el disco, pero los no compactos son compatibles con las anteriores versiones de FoxPro.
La razón de que existan dos método para indexar se fundamenta en la compatibilidad con versiones anteriores y con otros paquetes (dBase, Clipper, etc). Si su caso es compartir recursos con otras versiones del paquete o con otro tipo de manejadores de bases de datos será conveniente que utilice este tipo de índice no compactos, aún cuando los compactos y los CDX son mas eficientes en su uso y en cuanto a los recursos que necesitan para trabajar.

3.2.2.1 Creación de índices IDX

Para crear índices IDX desde pantalla el sistema nos mostrará la misma que los índices CDX con la modificación que nos dará entrada al nombre del archivo.

Como se puede ver aquí, crearemos un índice IDX compacto llamado Clave que posee como llave el rfc.



Si se deseara crear este tipo de índices desde la ventana de command o desde programa bastaría con teclear lo siguiente:

INDEX ON <campo llave> TO <nombre del índice> COMPACT

En el caso de tener un IDX compacto para el ejemplo anterior quedaría:

INDEX ON RFC TO CLAVE

3.2.2.2 Manipulación de los diferentes IDX

Cuando una misma base de datos contiene diferentes IDX se podrá ir alternando entre ellos indistintamente, ya que solo uno estará activo, para ello utilizaremos el comando

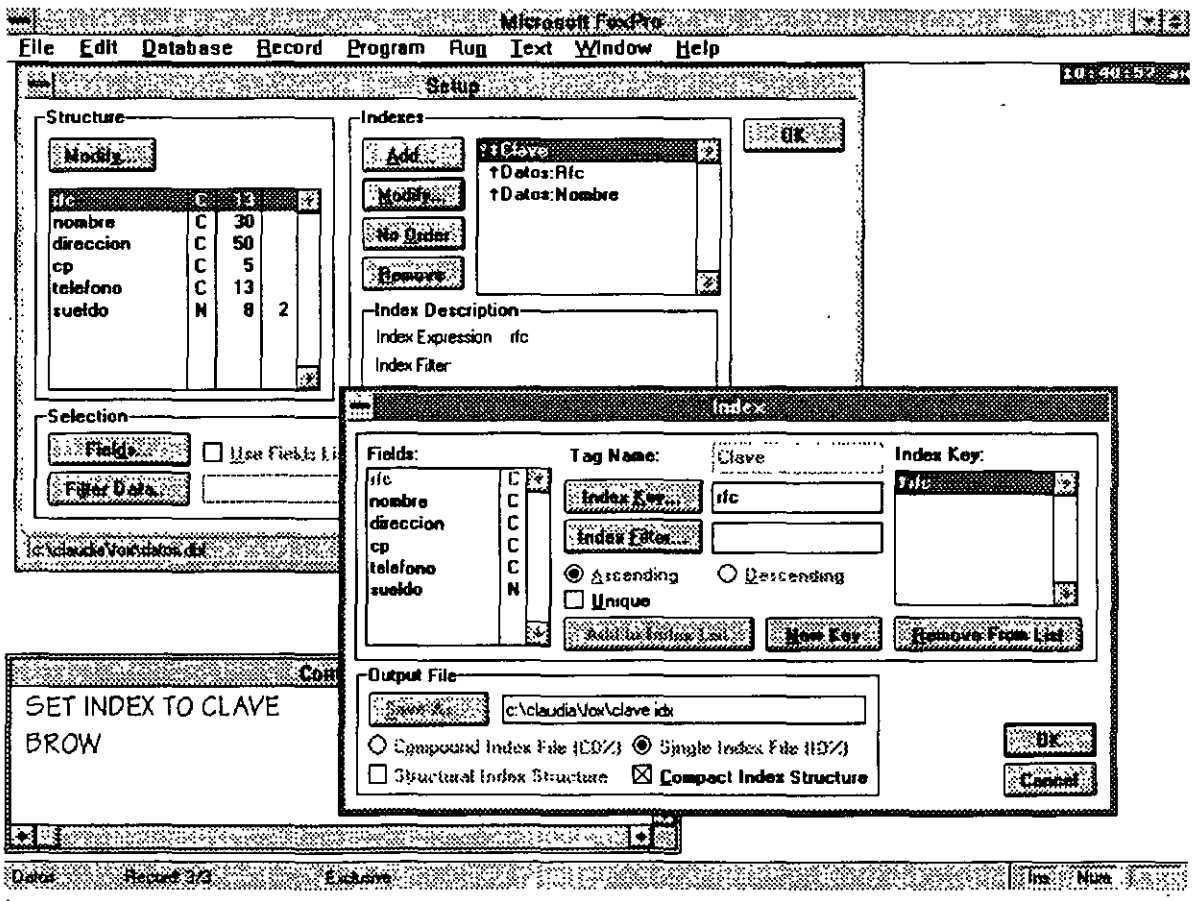
SET INDEX TO <nombre del índice>

En este caso tendríamos

SET INDEX TO CLAVE

3.3 Modificación de índices

Para el caso de que un índice no esté funcionando como se pensaba y se deba de modificar (cualquiera que sea su tipo) se seguirá el mismo procedimiento por menú Database - Setup y tendremos la misma pantalla que la de modificación de base de datos, solo que en este caso utilizaremos la parte derecha que es mostrada en la pantalla donde se nos muestra la lista de índices existentes.



En esta pantalla como vemos tendremos cuatro botones pegados al lado izquierdo de la misma :

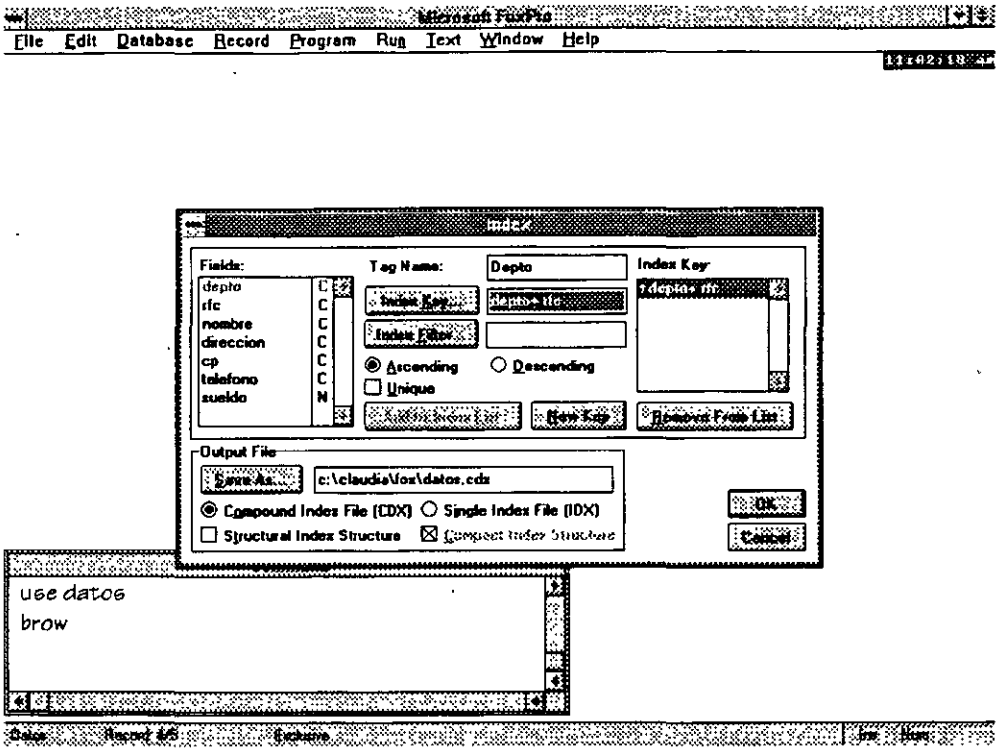
Add: Crear un nuevo índice
Modify: Modificar el índice seleccionado
Set order/No order: Tomar este índice como el activo o desactivarlo
Remove: Eliminar este índice.

En la ventana vemos tres nombres de índices Clave, Datos:rfc y Datos:nombre, este tipo de nomenclatura nos indica que el primero es un índice IDX y los dos siguientes son el índice DATOS.CDX y sus tags RFC y NOMBRE. Cuando aparezca una llave pequeña al lado izquierdo de algún índice se nos estará indicando que es la llave activa, apareciendo al mismo tiempo en el botón de ordenación No order, si deseamos desactivarlo bastará con oprimir éste botón que cambiará a Set order.

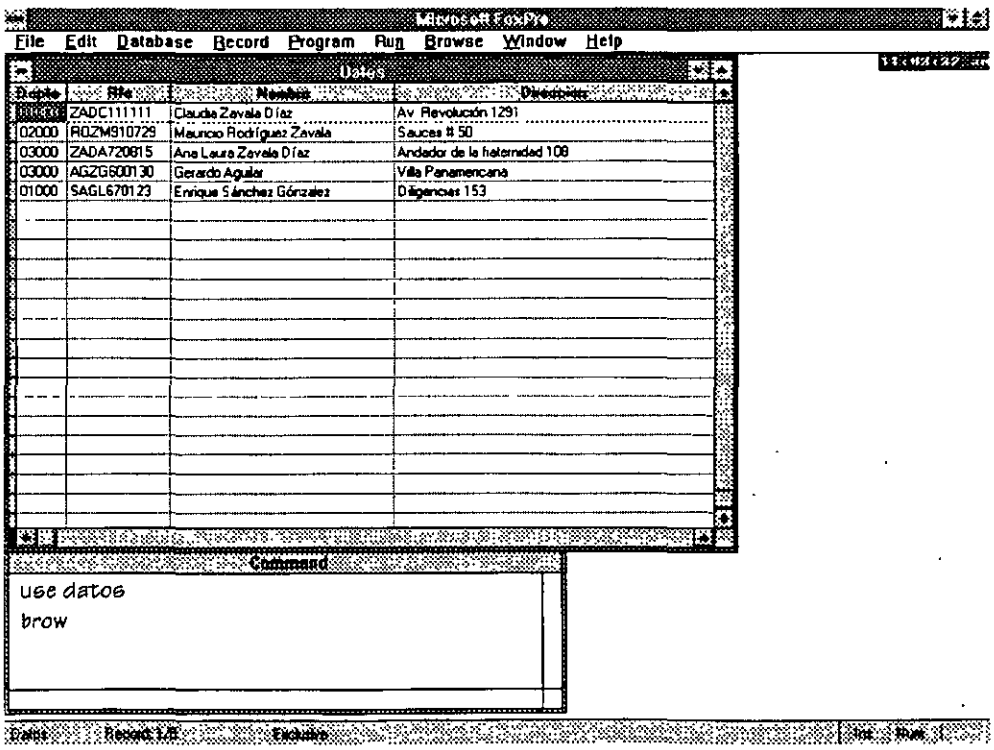
Cuando es necesario modificar la definición de un índice se oprimirá el botón Modify apareciendo la misma pantalla que manejamos cuando creamos los índices. Una vez que se han hecho las modificaciones requeridas se oprime Ok y Ok. Esto reindexará la base de datos.

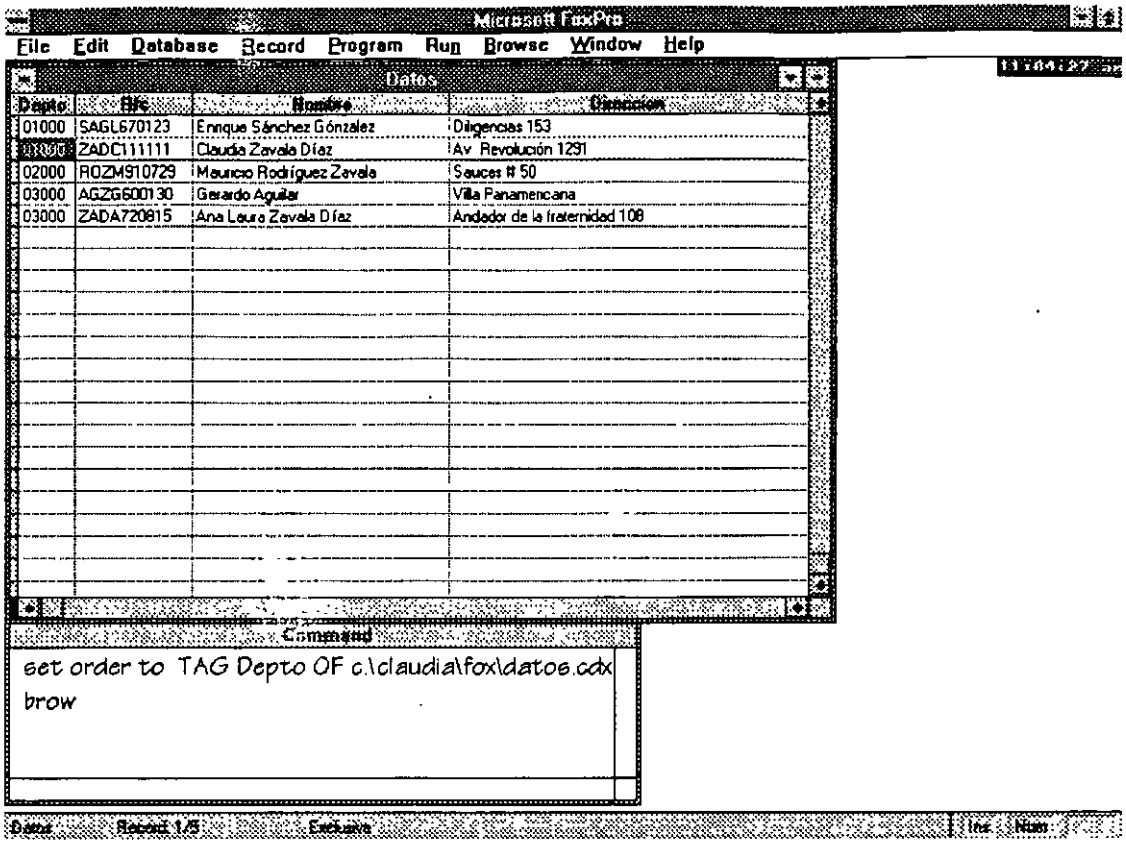
3.4 Índices combinados

Muchas veces es necesario que algún índice se encuentre formado por mas de un campo. Por ejemplo deseamos ordenar a todos nuestros empleados por su número de departamento pero a su vez por número de empleado, de tal forma que tuviéramos una referencia rápida a ellos por el lugar donde trabajan y su clave de empleado.
Para formar este tipo de índices independientemente del tipo de índice que se elija lo que se hará es concatenar o sumar ambas claves, pero se debe de tener en cuenta algo importante EL TIPO DE LOS DATOS A COMBINAR, esto es, no es posible formar un índice de el rfc con la fecha de ingreso, siendo el primero carácter y el segundo fecha, lo que se deberá de hacer será transformar a un mismo tipo de dato, todo a fecha o todo a carácter. Lo mismo sucederá si es numérico y carácter o numérico y fecha. Mas adelante veremos los comandos existentes para este tipo de transformaciones.
A continuación mostramos una base de datos indexada por No. de depto y rfc.



Teniendo el siguiente orden en la base de datos, primero en su forma natural y después en su forma indexada por los dos campos.





Nótese como muestra en primer lugar un orden por departamento y cada departamento a su vez se encuentra ordenado por RFC de cada empleado.

Nota: Para el caso de crear índices combinados con dos claves de tipo numérico se recomienda convertirlos a caracter y concatenarlos, debido a que en su forma numérica puede generar combinaciones similares con números diferentes, por ejemplo:

Clave= 1
Departamento=55
indice=clave + departamento // indice=155
indice=155

Clave= 15
Deparatamento=5 // indice =155

3.5 Ejercicio

Como ejercicio de estos dos últimos temas se tendrá lo siguiente:

1. Crear cuatro bases de datos con las siguientes estructuras:

DATOS_E

NO_EMPL	C	20	
DEPTO	C	5	
CATEGORI	C	5	
FECHA_ING	D	8	
TURNO	C	1	
COMENTA	M	8	
FOTO	G	8	
VIGENTE	L	1	
HORAS	N	2	0

DEPTOS

DEPTO	C	5	
DESCRIP	C	30	

CATEGORIA

CATEGORI	C	5	
DESCRIP	C	30	
SUELDO_H	N	6	2

DATOS_G

NO_EMPL	C	5	
NOMBRE	C	30	
CALLE	C	25	
COLONIA	C	30	
CP	C	5	
DELEGACION	C	20	
TELEFONO	C	15	

2. Crear lo siguientes índices
en Datos_e:
Un índice compuesto estructural con tres tags por número del empleado, por departamento, por departamento y número y por categoría y número del empleado
Indexar departamentos

Indexar categoría

en Datos_g:
Indexar con cdx por número del empleado y dos idx por nombre y cp.

3. Cerrar la bases de datos

4. Abrir Datos_e y contestar:
- ¿Qué índices están abiertos?
- ¿Son todos los creados?
- Si no lo son ¿por qué no están abiertos todos?, si lo son te falta terminar lo anterior

5. Abrirlos todos
- ¿Cuál está activo?

6. Activar el tag no_empl y visualizar la base de datos

7. Activar el tag compuesto y visualizar la base de datos
- ¿Encuentras alguna diferencia entre ambas vistas de la misma base de datos?
- ¿Cuáles?

8. Desactiva todos los índices y visualiza la base
- ¿Como se encuentra tu información?

9. Abre la base Datos_g y anexa el campo RFC tipo de dato caracter de 13 que se encuentre después del nombre del empleado.

10. Indexa por este campo con CDX no estructural y filtrando a los que su número de empleado sea menor a "0015"

11. Visualiza tu base de datos
- ¿Se encuentran todos tus registros?

12. Cambia tu índice al idx por número de empleado y visualiza
- ¿Cuál es la diferencia con el anterior?

4. Programación

4.1 Creación de programas

Definiremos como programas a las aplicaciones creadas por el usuario con un fin determinado, en este caso utilizando como lenguaje el propio del manejador de bases de datos Foxpro.
Para la creación de programas debemos de tomar en cuenta los siguientes pasos:

- Poseer una idea clara del problema
- Definir las entradas y salidas que tendrá nuestro programa

Para ello nos ayudaremos de diversas herramientas que proporcionará el lenguaje (comandos, funciones, estructuras lógicas, etc).
La forma óptima de crear un programa y utilizando como regla general la secuencia siguiente tendremos:

- Variables de ambiente
- Inicialización de variables a utilizar
- Abrir bases de datos e índices
- Si se requiere, estructuras lógicas que engloben una serie de instrucciones
- Regreso del programa a otro por el fue llamado o fin

Para iniciar explicaremos las estructuras lógicas .

4.2 Estructuras lógicas

Entenderemos las estructuras como figuras que nos permitirán tomar decisiones o tener a una serie de instrucciones dentro de un ciclo de repetición por un determinado número de veces. Esto facilitará la tarea del programador ya que le permitirá reducir código, controlar su programa y las entradas y salidas de este. A continuación veremos

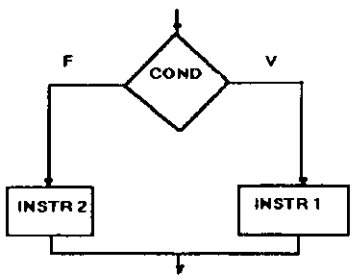
las figuras o estructuras lógicas con que cuenta Foxpro para después aplicarlas a nuestros programas.

4.2.1 IF- THEN- ELSE- ENDIF

La estructura IF se utilizará cuando el programa necesite tomar alguna decisión. En este caso se utilizará una comparación entre dos o más variables y dependiendo del resultado se llevarán a cabo cierto bloque de instrucciones. El valor que arroja la comparación resulta un valor de tipo booleano, por lo que si la comparación resulta verdadera (T) realizará las instrucciones que corresponde a la parte positiva (las instrucciones seguidas al THEN) y si el resultado fue falso (F) se ejecutarán las instrucciones de la parte negativa (instrucciones que se encuentran después del ELSE)

La sintaxis es la siguiente:

```
IF <condición> THEN
   instrucciones 1
ELSE
   instrucciones 2
ENDIF
```



El grupo de instrucciones 1 se realizarán cuando el valor de la condición sea verdadera de lo contrario realizará el conjunto de instrucciones 2.
Es importante hacer notar que sólo se realizará un bloque de instrucciones, nunca realizará ambas, este tipo de estructuras no es cíclico, por lo que solo se realizará una sola vez.

Un ejemplo de ello es el resultado de una operación en este caso en cuanto a si el valor obtenido es menor o mayor a cero:

```
A=F/H
IF A<0 THEN
   LETRERO="NUMERO DECIMAL"
ELSE
   LETRERO="NUMERO ENTERO"
ENDIF
```

4.2.2 FOR-NEXT

La estructura For es una estructura cíclica dependiente de una variable de tipo contador que permitirá realizar una serie de instrucciones mientras ésta no llegue a su límite, el desarrollador determinará el valor inicial de la variable y el valor final que deberá alcanzar, así como el valor de los incrementos (por default es 1).
El ciclo por si mismo irá incrementando a la variable contador sin que el usuario se preocupe por ello. Para el caso de que en lugar de incrementar se decremente el valor de los incrementos deberán de ser con signo negativo, así mismo el valor inicial deberá de ser mayor al final. En el momento que el contador llegué a su límite el ciclo terminará.

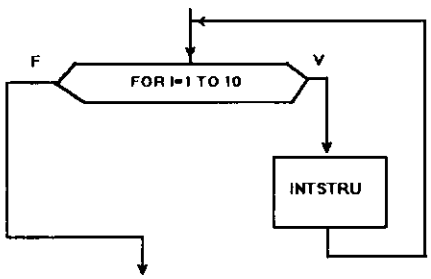
La sintaxis es la siguiente:

```
FOR contador=valor_inic TO valor_fin STEP #no_incrementos
    instrucciones
NEXT
```

Por ejemplo si deseáramos que un número se sumara 500 veces donde el valor inicial de la cuenta sería 1 su valor a alcanzar 1000 pero incrementando a la variable contador (I) de dos en dos se tendría lo siguiente:

```
FOR I=1 to 1000 STEP 2
   suma=suma+I
NEXT
```

A esta estructura lógica la podríamos identificar mediante la siguiente figura:



Como se puede ver en la figura el grupo de instrucciones seguirá siendo ejecutado mientras el contador no llegue al límite estipulado por el desarrollador.

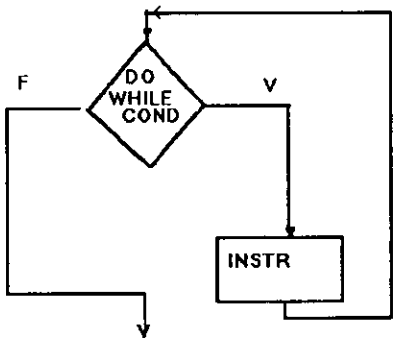
4.2.3 DO WHILE - ENDDO

El Do while es una estructura cíclica que realizará una serie de instrucciones mientras la condición que se evalúe resulte verdadera ya que el caso de ser falsa el ciclo se romperá y continuará con la

instrucción inmediata siguiente. En este tipo de estructura el número de interacciones que se realizarán no poseen un límite como en el caso del FOR, un detalle importante es que cuando se desee iterar un determinado número de veces utilizando esta estructura el valor que determinemos como contador deberá ser incrementado por alguna instrucción incluida dentro del ciclo ya que el While por sí solo no lo hará. Otro punto a considerar es el asegurarnos que siempre existirá una condición en algún momento determinado que haga falso el resultado de la comparación que rige al While, de lo contrario nunca saldrá del ciclo (mejor conocido como loop).

La sintaxis es la siguiente:

```
DO WHILE <condición>
   instrucciones
ENDDO
```



Tomaremos como ejemplo el mismo descrito en la instrucción For:

```
I=1
DO WHILE I<1000
     SUMA=SUMA+I
     I=I+1
ENDDO
```

En este caso el número de iteraciones no está controlado y se tendrá que ir incrementando a la variable contador.
Esta estructura se usará mas comúnmente para recorrer los elementos de una base de datos, teniendo siempre en consideración el ir incrementando al apuntador de registro de la base de datos.

4.2.4 DO CASE - ENDCASE

En el caso de la estructura CASE podría compararse con una serie de IFs anidados ya que su funcionamientoes semejante. También en este caso solo realizará un bloque de instrucciones a la vez y no es una estructura cíclica, en esta estructura tendremos muchas opciones a elegir y dependiendo del valor que tome la variable de control se realizará uno u otro bloque de instrucciones.

La sintaxis es la siguiente

```
DO CASE
    CASE <var_opción>= valor_a
        instrucción 1
    CASE <var_opción>=valor_b
        instrucción 2
    CASE  <var_opción>=valor_c
        instrucción 3
    OTHERWISE
        instrucción 4
ENDCASE
```

El Otherwise funciona cuando ninguno de los valores leídos resultó existir en la lista de opciones.
Este tipo de estructura comúnmente es utilizada para la llamada a diferentes subrutinas dependiendo de las opciones seleccionadas en algún menú o del resultado de alguna operación.

Un ejemplo de ello es lo siguiente:

```
@ 10, 01 say "Teclea lo opción deseada:" get opción
read
DO CASE
    CASE opcion=1
        @ 12,02 say "ALTAS"
    CASE opcion=2
        @ 12,02 say "BAJAS"
    CASE opcion=3
        @ 12,02 say "MODIFICACIONES"
    CASE opcion=4
        @ 12,02 say "CONSULTAS"
    OTHERWISE
        @ 12,02 SAY "SALIR"
ENDDO
```

Hasta aquí las estructuras vistas son utilizadas en todos los manejadores de bases de datos y en algunos lenguajes, quizá podría cambiar un poco la sintaxis pero la forma de manipular a las instrucciones y la forma de la estructura seguirá siendo la misma. En FoxPro tenemos una estructura más.

4.2.5 SCAN - ENDSCAN

Este tipo de estructura se utiliza para recorrer a los elementos de una base de datos.
Dentro de sus características más importantes se encuentra el hecho de que no se tendrá que incrementar a apuntador de base de datos ya que la estructura por si sola lo hará y recorrerá de inicio a fin la base de datos, siendo mas veloz que la estructura While. Una característica más será la de poder incluir condiciones en la

estructura, esto es poder filtrar solo determinada información con características seleccionadas.
La sintaxis de esta estructura es la siguiente:

```
SCAN <WHILE> | <FOR> <condiciones>
   <instrucciones>
ENDSCAN
```

Por ejemplo si quisiéramos leer y desplegar todos los elementos de una base de datos tendríamos:

```
USE DATOS_G
SCAN
  @ 1,1 SAY NO_EMPLEADO
  @ 2,1 SAY NOMBRE
ENDSCAN
```

Pero si sólo necesitáramos a los que su número de empleado fuera menor a "00100" tendríamos:

```
USE DATOS_G
SCAN WHILE NO_EMPLEADO<"00100"
  @ 1,1 SAY NO_EMPLEADO
  @ 2,1 SAY NOMBRE
ENDSCAN
```

o si necesitáramos solamente a los empleados de un departamento

```
USE DATOS_E
SCAN FOR DEPTO="5000"
  @ 1,1 SAY NO_EMPLEADO
  @ 2,1 SAY NOMBRE
ENDSCAN
```

Para el caso de manejar filtros será útil trabajar con la base de datos indexada, ya que esto nos asegura tener la información ordenada. Es

conveniente hacer notar que se deberá de llevar el apuntador de la base hasta el primer elemento que cumpla con la condición establecida en el filtro antes de ejecutar la estructura SCAN, de lo contrario como el primer elemento no cumple la condición terminará. El ejemplo anterior quedaría:

```
USE DATOS_E INDEX DATOS_E
DEPARTA="5000"
SEEK  DEPARTA
SCAN FOR DEPTO="5000"
  @ 1,1 SAY NO_EMPLEADO
  @ 2,1 SAY NOMBRE
ENDSCAN
```

Esto nos asegurará iniciar la lectura de la base de datos en un registro válido, además de darnos velocidad en el manejo de la información.

4.3 Variables de ambiente

Son todas aquellas variables que determinan el ámbito o ambiente de trabajo en el que se encontrará el sistema, algunos de ellos son:

Set alternate to <archivo.txt>	Graba en un archivo de texto toda la actividad habida en la pantalla (instrucciones y resultados)
Set alternate on/off	Activar el grabado en el archivo y de deberá cerrar

Un ejemplo de lo anterior será el siguiente

```
SET ALTERNATE TO SESION.TXT
SET ALTERNATE ON
USE DATOS_G
```

```
DISPLAY STRUCT
USE DATOS_E
SEEK "00015"
?RECNO()
SET ALTERNATE OFF
CLOSE ALTERNATE
```

Set bell on/off	Elimina o activa el sonido de la campana
Set border to [single, double, panel, none]	Define la cadena de bordes de los recuadros correspondientes a los menús y ventanas
Set century on/off	Modifica el formato de fecha mostrando el año con 4 dígitos.
Set color to <colores>	Define un ámbito de colores (se verá mas adelante los códigos)
Set confirm on/off	Requiere de un <Enter> para introducir un dato tecleado
Set console on/off	Muestra u oculta en la pantalla los comandos ejecutados en el programa
Set cursor on/off	Apaga o enciende la visualización del cursor
Set decimals to [número]	Visualiza un número determinado de decimales para las fracciones numéricas
Set default to [unidad de disco y/o directorio]	Especifica una unidad de disco o directorio que se utilizará por default
Set delete on/off	Encendido servirá para filtrar aquellos registros que se encuentren marcados para ser borrados, apagado ignora los registros que han sido marcados para borrar.
Set delimiters on/off	Apaga o enciende los delimitadores
Set delimiters to "[]"	Define cuáles será los delimitadores
Set device to [screen, printer] Set device to file <archivo.txt>	Direcciona la salida de un programa a una unidad especifica como lo es la pantalla, la impresora o algún archivo de resultados.

Set echo on/off	Observa o elimina el escribir en la ventana de command las instrucciones de comandos que se realizan por menú
Set escape on/off	Permite o impide la salida de una instrucción al oprimirse <Esc>
Set fixed on/off	Fija el número de decimales a ser desplegado dependiendo del set decimals
Set intensity on/off	Apaga o enciende la intensidad en los gets .
Set margin to <número>	Fija el número de espacio de margen izquierdo cuando se envía a impresora.
Set memowidth to <número>	Determina el ancho de visualización de un campo memo
Set scoreboard on/off	Controla los mensajes que se despliegan a pantalla después de una instrucción READ
Set window of memo to <nombre>	Define una ventana específica como la de despliegue para un campo memo

Set date: Definirá el formato de fecha a utilizarse

ansi	aa.mm.dd
italian	dd-mm-aa
british	dd/mm/aa
french	dd/mm/aa
german	dd.mm.aa
japan	aa/mm/dd
american	mm/dd/aa
usa	mm-dd-aa

4.4 Operadores

4.4.1 Operadores matemáticos

+	Suma
-	Resta
*	Multiplicación
**	Exponenciación
/	División

4.4.2 Operadores relacionales

.and.	Conjunción
.or.	Disyunción
.not.	Negación

4.4.3 Operadores lógicos

<	Menor que
<=	Menor o igual
>	Mayor que
>=	Mayor o igual
<> #	Diferente
=	Igual

4.4.4 Otros operadores

( )	Indicador de función o agrupamiento
{.../.../...}	Fecha nula
&	Macrosustitución
? | ??	Desplegado de información
@	Pase de información por referencia

4.5 Tipos de variables

Manejaremos en este apartado los tipos de variables tomando en cuenta su duración o periodo de vida.

Públicas: Son válidas en todos los programas siendo modificables en cualquier parte del sistema. Se liberan cuando se libera la memoria. Se declaran en cualquier parte del programa.

PUBLIC <lista de variables>

Privadas: Tienen vida mientras está vigente el programa en el que fueron creadas. Se declaran al principio del programa. Cuando en un programa se declara como privadas a una lista de variables y existe una llamada a alguna subrutina dentro de éste, en el momento de cambiar el control a la subrutinas, los valores privados se ocultarán, por lo que si en este programa existe una variable con el mismo nombre el valor anterior se sigue conservando ya que en el momento de salir del subprograma se liberan las variables definidas en éste y reaparecen las variables privadas con sus valores.

PRIVATE <lista de variables>

4.6 Inicialización de variables

La inicialización de las variables consiste en darles un valor con el que iniciarán el programa, cabe aclarar que de no inicializarlas el programa marcará error al encontrarlas por primera vez. Dependiendo del tipo de la variable será la forma a inicalizar.

Caracter

Var_char=SPACE(#)	Se inicializa con un número determinado de espacios en blanco
STORE " " TO var_char	Grabará en la variable caracter espacios en blanco, tantos como se le definan

Numéricas

var_num=0	Se inicializa con cero
STORE 0 TO var_num	Se almacena un cero en la variable

Lógicas

var_log=.T. o .F	Solo acepta valores de verdadero o falso

.Fecha

var_fech=date()	Es posible inicializar con la fecha del dia
var_fech=CTOD(" / / ")	Se inicializa en blanco ayudándose de una función especial
{../../..}	Fecha nula

4.7 Manejo de bases e índices dentro del programa

Para la utilización de bases de datos dentro de un programa será necesario seleccionar áreas de trabajo, para ello se utilizará el comando SELECT y un número, él cuál será la etiqueta de la base de datos, de tal

modo que se hará referencia a ésta por medio solo de la etiqueta. Una vez elegida el área de trabajo en la línea inmediata inferior se determinará la base de datos que poseerá dicha etiqueta (para el caso de que no se encuentre previamente abierta, de lo contrario se hará sólo referencia al número de etiqueta), cuando la base de datos posee índice será necesario indicarle que lo(s) abra en la misma área de trabajo.

Por ejemplo:

```
SELECT 1
USE DATOS_G
SELECT 2
USE DATOS_E
```

En el ejemplo anterior sólo se abrieron las bases de datos asignándolas a un área determinada de trabajo. Ahora para hacer referencia a ellas ya abiertas en FoxPro tendremos variedad como se verá a continuación:

```
SELECT 1
SELECT 2
```

En este caso se hará referencia al número de la etiqueta de cada base de datos.

```
SELECT DATOS_G
SELECT DATOS_E
```

Aquí utilizamos como apuntador el nombre de la base de datos, esto resulta práctico cuando tenemos que hacer que pasar como parámetro el nombre de alguna base de datos.

```
SELECT A
SELECT B
```

En esta forma estaremos seleccionando el área con letras, de la misma forma que lo hacemos con números en donde el A corresponde a 1, B a 2

y así sucesivamente, por lo que select A llamará a DATOS_G y B a DATOS_E.

Cuando utilizamos índices dependiendo de su naturaleza se deberán de incluir en la asignación de la base a la etiqueta o no.

Para el caso de tener índices CDX, como se dijo anteriormente se abren en el momento de abrir la base por lo que no será necesario indicárselo al momento de la asignación. Cuando el que se quiere abrir junto con la base de datos es el índice IDX, será necesario indicarlo.

Por ejemplo:

En el caso siguiente la base de datos se encuentra indexada por rfc con un índice CDX.

```
SELECT 1
USE DATOS_G
```

si visualizáramos la base de datos nos la mostraría la base en su forma natural. Para activar el índice utilizaríamos el comando visto anteriormente.

```
SELECT 1
USE DATOS_G ORDER TAG RFC OF DATOS_G
```

Para el caso de haberlo abierto sin indicarle el orden a tomar se tendría que hacer posteriormente con el comando:

```
SET ORDER TO RFC
```

Cuando se trata de los índice IDX si se deberán de abrir al tiempo de asignar el área de trabajo, se tiene:

```
SELECT 1
USE DATOS_E INDEX DATOS_E
```

Si se desea hacer posteriormente se utilizará el comando:

```
SELECT 1
USE DATOS_E
SET INDEX TO DATOS_E
```

Para poder tener una idea mas clara de este manejo se presentan unos ejemplos a continuación:

* Selecciona en la primera área a DATOS_E abriendo su índice CDX DATOS_E, lo que hará indexar la base sobre el campo NO_EMPL.

```
SELECT 1
USE  DATOS_E ORDER TAG NO_EMPL OF DATOS_E
```

* Selecciona en la segunda área a DATOS_G y su índice RFC

```
SELECT 2
USE DATOS_G INDEX RFC
```

* Se activa el área 1 (la base DATOS_E) y la recorremos con la estructura lógica SCAN desplegando tres campos a pantalla

```
SELECT 1
SCAN
  ? NO_EMPL,RFC,DEPTO
ENDSCAN
```

* Seleccionamos el área 2 y hacemos lo mismo

```
SELECT 2
SCAN
  ? NO_EMPL,NOMBRE,TELEFONO
ENDSCAN
RETURN                                  // Termina
```

4.8 Ejercicio

1. Crear un nuevo programa llamado muestra.prg por medio del menú.

2. Define tus variables de ambiente, incluyendo aquella donde se puede mandar a un archivo todas las salidas de la pantalla a un archivo.

3. Abre las bases de datos datos_g y datos_e ambas con sus índices dejando activo en la primera el índice de número de empleado y en la segunda el nombre.

4. Despliega ambas bases de datos a pantalla sólo los campos en la primera no_empl, depto, catego y en la segunda no_empl, nombre y teléfono.

5. Despliega de datos_g la base de datos ahora ordenada por depto y número de empleado en este caso los mismos campos que en el anterior.

6. Cierra el llamado a salida a papel

7. Termina

5. Funciones y comandos

A continuación se verán las funciones y comandos más utilizados aún cuando el compilador cuente con un set de instrucciones más amplio.

@... SAY GET

Sintaxis

@ ren,col SAY "<texto>"
PICTURE <formato pict>
FUNCTION <tipo de función a realizar>
COLOR SCHEME <cadena de colores>

@ren,col GET <variable>
PICTURE<formato pict>
FUNCTION <tipo de función>
STYLE <tipo de estilo>
FONT <tipo de font>, <tamaño>
DEFAULT <valor por omisión>
MESSAGE <mensaje>
RANGE <rango>
SIZE <tamaño>
VALID <condición a validar>
ERROR <en caso de error>
WHEN <condición>
COLOR RGB <colores>

Definición

La única función del comando SAY, es desplegar un letrero o variable en pantalla dependiendo de las coordenadas especificadas.
Para el caso de la instrucción GET, ésta servirá para darle al usuario la opción de proporcionar información para alimentar al sistema,

siempre deberá de ir acompañado de la instrucción READ al final de las series de GET que se utilicen en el programa ya que de esta forma los datos proporcionados realmente se instancian en las variables.

Opciones

PICTURE

A	SOLO LETRAS
L	VALORES LÓGICOS
N	NÚMEROS Y LETRAS
Y	YES o NO
X	CUALQUIER SÍMBOLO
9	SÓLO DÍGITOS
#	LETRAS, ESPACIOS Y SIGNOS
!	SÓLO MAYÚSCULAS
.	POSICIÓN INICIAL
,	SEPARADOR DE MILES
$	CURRENCY
*	RELLENA CON ASTERÍSCOS
@	TODA LA LONGITUD DEL CAMPO

FUNCIÓN

A	LETRAS
B	JUSTIFICA A LA IZQUIERDA
D	UTILIZA EL FORMATO DE FECHA ESPECIFICADO CON SET DATE
E	EDITA LA FECHA CON FORMATO BRITÁNICO
I	CENTRA EL CAMPO
J	JUSTIFICA A LA DERECHA
L	DESPLIEGA SIN CEROS A LA IZQUIERDA
S (N)	NÚMERO DE ESPACIOS A DESPLEGAR
Z	DESPLIEGA BLANCO SI LA VARIABLE ES CERO
!	TODO EN MAYÚSCULAS
^	NÚMEROS EN NOTACIÓN CIENTÍFICA
$	CURRENCY

ESTILO

B BOLD
C CONDENSADA
E EXTENDIDA
I ITÁLICA
N NORMAL
O OUTLINE
Q OPACA
S SOMBREADA
- TACHADA
T TRANSPARENTE
U SUBRAYADA

COLOR RGB

BLANCO	255.255,255
NEGRO	0,0,0
GRIS	192,192,192
GRIS OBSCURO	128,128,128
ROJO	255,0,0
ROJO OBSCURO	128,0,0
AMARILLO	255,255,0
AMARILLO OBSCURO	128,128,0
VERDE	0,255,0
VERDE OBSCURO	0,128,0
AZUL CIELO	0,255,255
AZUL CIELO OBSCURO	0,128,128
AZUL	0,0,255
AZUL OBSCURO	0,0,128
ROSA	255,0,255
ROSA OBSCURO	128,0,128
SIN COLOR	,,

En el caso de color RGB se determinaran dos tercias de números entre paréntesis que definirán, la primera lo escrito y las segundas el fondo.

EJEMPLO

```
NOMBRE="Aurora"
TEXTO="**********"
VALOR=1000.20
VALOR2=-500.50
T_DESC=SPACE(1)
USE FUNC
@01,01 SAY "NOMBRE :"+NOMBRE  COLOR RGB (0,128,0,128,128,128)
@02,01 SAY "SALARIO: "
@02,12 GET SALARIO PICTURE "999,999.99" RANGE 1,100000;
  WHEN  NOMBRE<>' ' FUNCTION  "I" ;
  FONT "DESDEMONA",12
@03,50 SAY "DESCUENTO: "
@03,12 GET VALOR2 PICTURE "999,999.99" VALID VALOR2>0;
        STYLE 'BI'
@04,05 SAY "TIPO DE DESCUENTO" GET T_DESC ;
        VALID T_DESC$"ABCD"
@06,01 SAY TEXTO PICTURE "!!!!!"
@07,01 SAY TEXTO PICTURE "@!"
READ
```

EJERCICIO

1. Crear un programa que se llame altas.prg donde se introduzcan datos a la base DATOS_E donde:
2. No se acepte el no_empl en blanco
3. La fecha de ingreso sea menor a la de hoy y color rojo
4. Horas no sea mayor a 40 ni menor a 20
5. El turno sólo pueda ser Matutino o Vespertino
6. En depto sólo se acepten números y el tipo de letra sea bold
7. Vigente sólo acepte valores booleanos
8. Categoría sea de color azul obscuro y gris con tipo de letra Foxprint de 12 puntos

@... BOX

Definición:

Visualiza en la pantalla una caja o ventana, siempre que se definan las 9 partes que lo componen. Para situar el marco, se incluyen las coordenadas de la esquina superior izquierda y la inferior derecha.

Sintaxis

@ren1,col1,ren2,col2 BOX "c1c2c3c4c5c6c7c8c9"

La cadena posee las sig. características

c1 : caracter de la esquina superior izquierda
c2: caracter para la línea superior
c3: caracter para la esquina superior derecha
c4: caracter para la línea derecha
c5: caracter para la esquina inferior derecha
c6: caracter para la línea inferior
c7: caracter para la esquina inferior izquierda
c8: caracter para la línea izquierda
c9: caracter de relleno

Ejemplo

```
CLEAR
@05,10,10,20 box "12345678"
```

Dibujará una caja de esta forma

```
122222222223
8           4
8           4
8           4
8           4
7666666665
```

@.... TO

Definición

Dibuja una caja o un marco en la pantalla definiendo el tipo de bordes a utilizar (sencillos o dobles)

Sintaxis

@ ren1,col1 TO ren2,col2 [doublel panel] color <lista colores>

Ejemplo

@2,2 TO 10,25 DOUBLE COLOR N/BG

@ ... CLEAR TO

Definición

Limpia la pantalla dependiendo de las coordenadas indicadas

Sintaxis

@ ren1,col1 CLEAR TO ren2,col2

Ejemplo

@10,02 SAY "Letrero"
@15,02 CLEAR TO 15,78
@15,02 SAY "Teclea datos:" GET DATOS

ACTIVATE WINDOW

Definición

Muestra y activa una ventana. Cuando se activa una ventana hasta que no se desactive o se active una nueva las coordenadas sobre las que se trabajarán estarán con base en la nueva ventana que será el marco de referencia.

Sintaxis

ACTIVATE WINDOW <nombre de la ventana o lista de ventanas>

Ejemplo

```
CLEAR
DEFINE WINDOW VENTA1 FROM 4,4 TO 20,20
ACTIVATE WINDOW VENTA1
@1,1 SAY "AREA DE CAPTURA"
DEACTIVATE WINDOW VENTA1
RETURN
```

ALLTRIM()

Definición

Elimina espacios en blanco a la derecha y a la izquierda de una cadena de caracteres

Sintaxis

ALLTRIM(CADENA)

Ejemplo

```
CLEAR
USE DATOS_G
@2,2 SAY "NOMBRE : " GET NOMBRE1
READ
APPEND BLANK
REPLACE NOMBRE WITH ALLTRIM(NOMBRE1)
RETURN
```

APPEND BLANK

Definición

Agrega un registro en blanco al final de la base de datos que se encuentra activa

Sintaxis

APPEND BLANK

Ejemplo

```
USE FUNC
@01,01 SAY "NOMBRE :"+NOMBRE
@02,01 SAY "SALARIO: "
@02,12 GET SALARIO PICTURE "999,999.99" RANGE 1,100000;
        WHEN NOMBRE<>' '
@03,01 SAY "DESCUENTO: "
@03,12 GET VALOR2 PICTURE "999,999.99" VALID VALOR2>0
@06,01 SAY TEXTO PICTURE "!!!!!"
@07,01 SAY TEXTO PICTURE "@!"
@15,01 SAY DATE() PICTURE "@D"
READ
APPEND BLANK
REPLACE SAL   WITH SALARIO
REPLACE DESC  WITH VALOR2
```

APPEND FROM

Definición

Agrega datos de una base de datos a otra activa dependiendo o no de una cierta condición

Sintaxis

APPEND FROM<de donde> fields <campos> for <condición>

Ejemplo

En este ejemplo solo anexará los registros de datos_gr a nomina que contengan como clave la 5000

```
SELECT 1
USE DATOS_GR
SELECT 2
USE NOMINA
APPEND FROM DATOS_GR TO NOMINA FIELDS CLAVE,NOMBRE;
FOR CLAVE>'5000'
```

ASC()

Sintaxis

ASC("string")

Definición

Retorna el valor del código ASCII del primer caracter de la expresión de caracteres.

Ejemplo

```
VAR_NUM=ASC("A")
```

AT()

Definición

La función AT nos indicará la posición de la primera incidencia de una cadena de caracteres en otra.

Sintaxis

AT(<subcadena>,<cadena>) se asignará a una variable de tipo numérico

Ejemplo

```
CLEAR
CADENA="VAR1+VAR2="
NUMERO=0
NUMERO=AT("=",CADENA)
IF NUMERO<>0
  @1,1 SAY "Operación completa"
ELSE
  @1,1 SAY "Operación incompleta"
ENDIF
RETURN
```

AVERAGE

Definición

Obtiene el promedio de los campos numéricos de la base de datos y lo almacena en una variable.

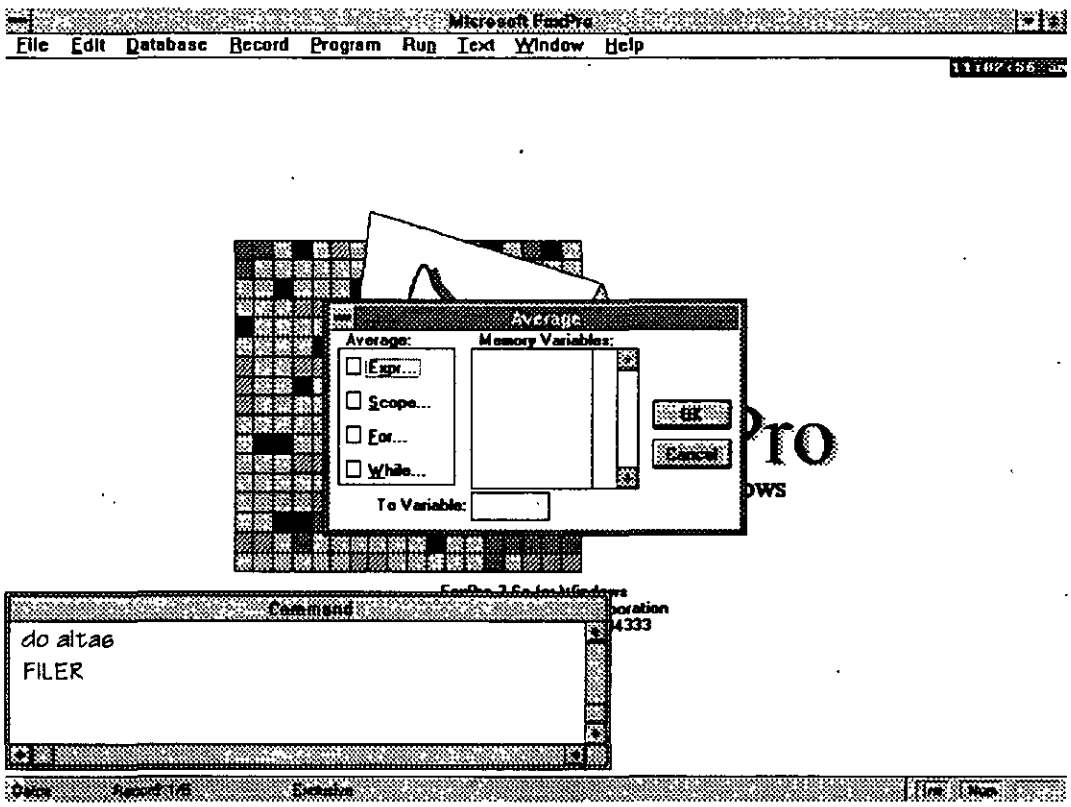
Sintaxis

AVERAGE <Lista de campos> [NEXT #]
[FOR <condición>] [WHILE <condición>]
TO <lista de variables>

Ejemplo

```
USE DATOS_G
PROM_SUEL=0
AVERAGE SUELDO  FOR NUM_DEPTO="1000" TO PROM_SUEL
```

Por medio del menú es posible realizar también este tipo de operaciones con las bases de datos, accesando al menú Database - Average tendremos:



Donde se nos preguntará sobre el campo a operar, cuantos elementos se tomarán en cuenta, si existe alguna condición while o for y en que variable se almacenará el resultado obtenido.

BOF()

Definición

Pregunta por el inicio de archivo, devolviendo un valor lógico.

Sintaxis

BOF()

Ejemplo

```
USE DAT_GRAL
IF BOF()
  @1,1 SAY "ARCHIVO VACIO"
ENDIF
```

BROWSE

Definición

Es el comando que le permitirá al usuario desplegar el contenido de la base de datos. Este se podrá manipular dependiendo de las características que nos permitirá switchear.

Sintaxis

BROWSE
[FIELDS <lista de campos a desplegar>]
[FOR <condición>]
[FONT <tipo de font>,<tamaño>]
[STYLE <estilo de letra>]
[FREEZE <campo>]
[NOAPPEND]
[NOCLEAR]
[NODELETE]

[NOEDIT | NOMODIFY]
[NOMENU]
[TITLE <título>]
[WHEN <condición>]
[WINDOW <nombre de la ventana>]
[COLOR <lista de colores>]

Ejemplo

```
USE DATOS_G
BROWSE FIELDS NUM_EMPL,NOMBRE,TELEFONO NOAPPEND
NODELETE;
FONT "Times New Roman",12
```

CDOW()

Definición

Retorna el nombre del dia de la semana correspondiente a la expresión de tipo fecha suministrada como argumento. Para una fecha en blanco retorna una cadena vacía.

Sintaxis

CDOW(fecha)

Ejemplo

```
FECHA=DATE()
DIA=CDOW(FECHA)
@1,1 "EL DIA DE HOY ES :" + DIA
```

CHR()

Definición

Convierte un no. ASCII en su caracter

Sintaxis

CHR(#)

Ejemplo

```
VAR_CHAR=CHR(5)
```

CLEAR

Definición

Comando que nos permitirá desde borrar la pantalla hasta limpiar las variables de memoria, pantallas, etc.

Sintaxis

CLEAR ALL | FILEDS | GETS | MACROS | MEMORY | MENUS | POPUPS | PROGRAM | PROMPT | READ | WINDOWS

Ejemplo

```
CLEAR
USE DATOS_G
DO WHILE .T.
 @1,1  SAY "NOMBRE : " GET NOMB
 @2,1 SAY "TELEFONO: " GET TELEF
 READ
 APPEND BLANK
 REPLACE NOMBRE WITH NOMB
 REPLACE TELEFONO WITH TELEF
 CLEAR GETS
ENDDO
```

CLOSE

Definición

Cierra bases de datos, índices, enviados de instrucciones a archivos mediante alternate.

Sintaxis

CLOSE ALTERNATE | DATABASE | INDEX

Ejemplo

```
USE DATOS_G INDEX DATOS_G
DO WHILE .T.
 @1,1  SAY "NOMBRE : " GET NOMB
 @2,1 SAY "TELEFONO: " GET TELEF
 READ
 APPEND BLANK
 REPLACE NOMBRE WITH NOMB
 REPLACE TELEFONO WITH TELEF
 CLEAR GETS
ENDDO
CLOSE DATABASE,INDEX
```

CMONTH()

Definición

Retorna el nombre del mes de la fecha suministrada como argumento.

Sintaxis

CMONTH(fecha)

Ejemplo

```
FECHA=DATE()
MES=CMONTH(FECHA)
@1,1 "EL MES ES: "+MES
```

COL()

Definición

Devuelve la coordenada de la columna donde actualmente se encuentra el cursor

Sintaxis

COL()

Ejemplo

```
LETRERO="TEXTO"
COLUMNA=COL()+15
@1,COLUMNA SAY LETRERO
```

CONTINUE

Definición

Continúa con una búsqueda realizada con el comando Locate

Sintaxis

CONTINUE

Ejemplo

```
SELECT 1
USE  ALMACEN
SELECT 2
USE VENTAS INDEX VENTAS
CONTINUA=.T.
DO WHILE CONTINUA
  @ 10,1 SAY "TECLEE LA CLAVE : " GET WCLAVE
  READ
  SELECT 1
  DO WHILE .NOT. EOF()
    LOCATE FOR CLAVE=WCLAVE
```

```
        @ 10,25 SAY DESCRIPCION
        CONTINUE
      ENDDO
    ENDDO
    RETURN
```

COPY FILE

Definición

Copia una base de datos en un archivo

Sintaxis

COPY FILE <arch fuente> TO <arch destino>

Ejemplo

```
COPY FILE ARCHIVO1 TO ARCHIVO2
```

COPY MEMO

Definición

Permite copiar el contenido de un campo memo en un archivo

Sintaxis

COPY MEMO <campo memo> TO <archivo> [ADDITIVE]

Ejemplo

```
USE DATOS_E
COPY MEMO OBSERVA TO OBSERVA.TXT ADDITIVE
```

COPY STRUCTURE

Definición

Copia la estructura de la base de datos activa a otra, si no existe la crea.

Sintaxis

COPY STRUCTURE fields <campos> TO <base de datos>

Ejemplo

```
CLEAR
USE DATOS_GR
COPY STRUCTURE TO  FIELDS CLAVE,NOMBRE TO DATOS_NOM
```

COPY STRUCTURED EXTENDED

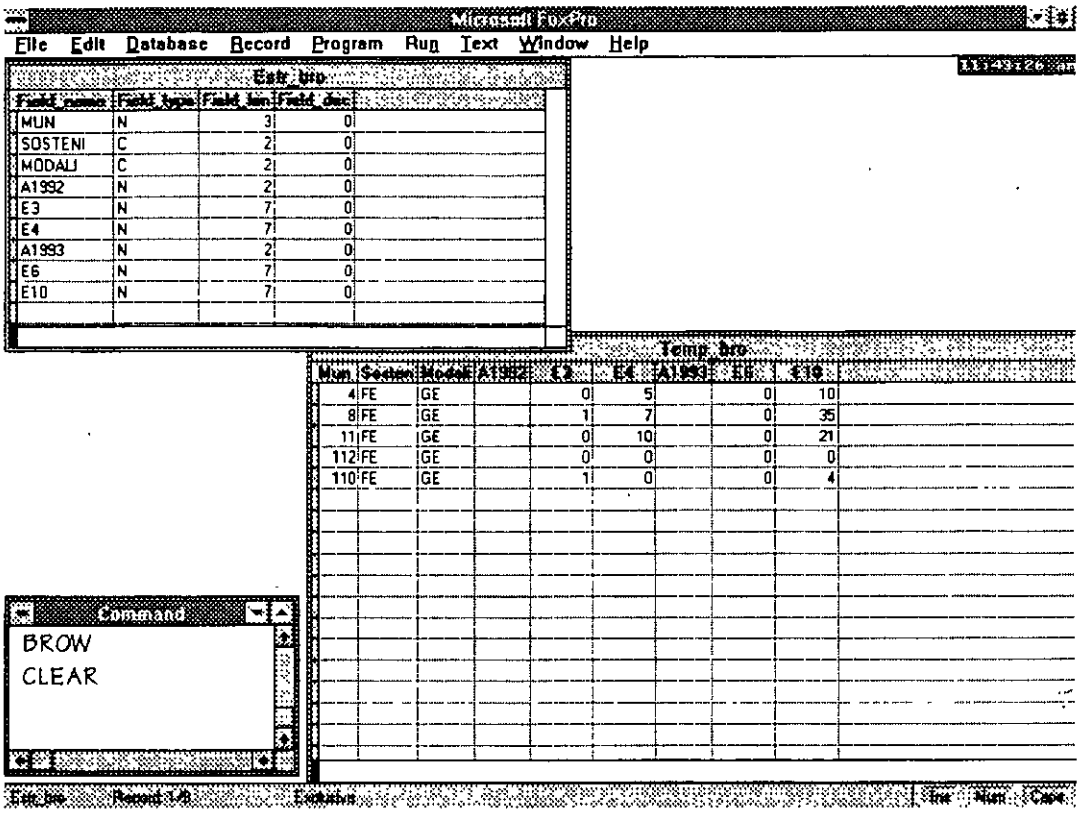
Definición

Esta instrucción es capaz de copiar en un registro de una base de datos el nombre (field_name), tipo (field_type), longitud(field_len) y los decimales (field_dec) que definirán la estructura de una base de datos.
Por lo que podríamos decir que es una base de datos que almacena la estructura de otra.

Sintaxis

COPY STRUCTURE EXTENDED TO <nueva base>

Ejemplo



En este caso la base de datos ESTR_BRO definirá la estructura de la base de datos TEMP_BRO como se puede observar. Esto nos permitirá tener bases de datos dinámicas.

COPY TO

Definición

Copia parcial o totalmente una base de datos a una nueva o a un archivo tipo ASCII con un determinado formato.

Sintaxis

COPY TO <nombre archivo>
[FIELDS <lista de campos>]
[NEXT #]
[FOR <condición>]
[WHILE <condición>]

[TYPE SDF | WK1 | WKS | WRI | WRK | XLS | DELIMITED WITH <caracter delimitador> | WITH BLANK | WITH TAB]

Ejemplo

```
USE DATOS_G INDEX DATOS_G
DO WHILE .T.
 @1,1 SAY "NOMBRE : " GET NOMB
 @2,1 SAY "TELEFONO: " GET TELEF
 READ
 APPEND BLANK
 REPLACE NOMBRE WITH NOMB
 REPLACE TELEFONO WITH TELEF
ENDDO
COPY TO BASE FIELD NOMBRE,TELEFONO TYPE XLS
CLOSE DATABASE,INDEX
```

Donde base es un archivo que contendrá la información de la base de datos de los campos seleccionados en formato de Excel.

COUNT

Definición

Cuenta los registros que cumplen con alguna condición determinada y asigna el resultado a una variable.

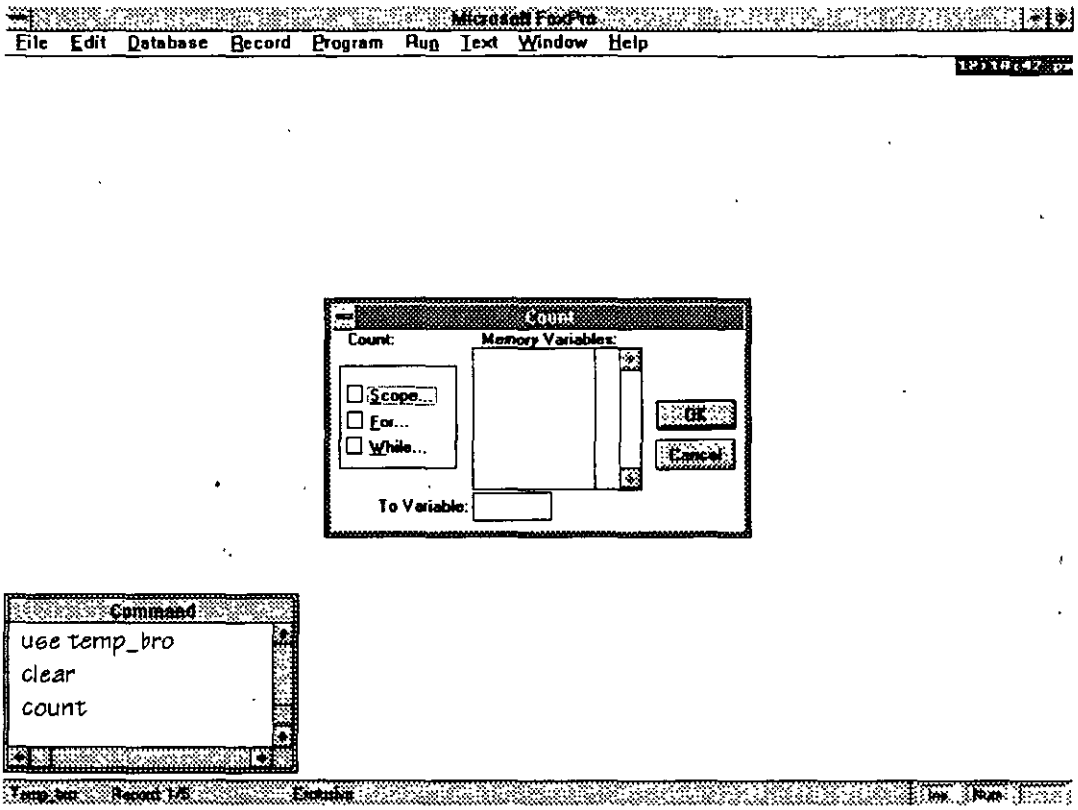
Sintaxis

COUNT [<límite>]
[FOR <condición>]
[WHILE <condición>]
TO <variable de memoria>

Ejemplo

```
USE DATOS_E INDEX DATOS_E
CUANTOS=0
COUNT FOR NUM_EMPLEADO<"1000" TO CUANTOS
```

Por medio de menú podríamos realizar esta misma operación llamando al menú Database-Count



CREATE FROM

Definición

Crea una base de datos que adopta la misma estructura que una obtenida mediante el COPY EXTENDED.

Sintaxis

CREATE FROM <base>

Ejemplo

```
USE DATOS_G
COPY STRUCTURE EXTENDED TO TEMPORAL
CREATE DATOS_G1 FROM TEMPORAL
```

CTOD()

Definición

Convierte una fecha que está almacenada en forma de una variable de caracteres hacia la variable de tipo fecha

Sintaxis

CTOD(string fecha)

Ejemplo

```
FECHA=CTOD(" / / ")
```

CURDIR()

Definición

Visualiza el directorio actual del DOS de una determinada unidad de disco. El argumento es un expresión de caracteres que debe estar entre comillas y contiene la letra de la unidad por investigarse. Si se omite la expresión retornará el contenido en la unidad y directorio actuales.

Sintaxis

CURDIR("drive:")

Ejemplo

```
@1,1 TO 20,78
@19,2  SAY CURDIR("C:")
```

DATE()

Definición

Fecha del sistema

Sintaxis

DATE()

Ejemplo

```
@ 1,1 SAY "FECHA: " +DTOC(DATE())
```

DAY()

Definición

Retorna el dia del mes (número) de la expresión suministrada como argumento

Sintaxis

DAY(fecha)

Ejemplo

```
@1,1 SAY "EL DIA DE HOY ES "+STR(DAY(FECHA))
```

DBF()

Definición

Regresa el nombre de la base en uso

Sintaxis

DBF()

Ejemplo

```
SELECT 1
USE DATOS_GR
? DBF()
```

DEFINE WINDOW

Definición

Crea una ventana para posteriormente trabajar con elllla con ACTIVATE WINDOW.

Sintaxis

```
DEFINE WINDOW  <nombre>
  FROM <ren1,col1> TO ren2,col2
  [FOOTER <pie de ventana>]
  [TITLE <título>]
  [DOBLE | PANEL | NONE | SYSTEM <cadena>]      forma del marco
  [CLOSE | NOCLOSE ]                  botón para cerrarla
  [FLOAT | NOFLOAT ]                  botón para moverla
  [GROW | NOGROW]                     botón para maximizarla
  [MINIMIZE]                          botón para minimizarla
  [ZOOM | NOZOOM]                     botón para el zoom
  [FILL <caracter de relleno>]
  [COLOR <lista de colores>]
```

Ejemplo

```
USE DATOS_G
DEFINE WINDOW VENTANA FROM 4,10 TO 10,55;
    TITLE "PANTALLA DE CAPTURA"
DEFINE WINDOW VENT1 FROM 12,10 TO 15,55;
    FOOTER "SEGUNDA VENTANA" MINIMIZE
ACTIVATE WINDOW VENTANA
@1,1 SAY "NOMBRE: " GET NOMBRE
ACTIVATE WINDOW VENT1
@1,1 SAY "TELEFONO: " GET TELEFONO
READ CYCLE
```

DELETE

Definición

Borra campos de una base dependiendo o no de una condición

Sintaxis

DELETE <scope> [FOR <condición>] [WHILE <condición>]

Ejemplo

```
DELETE ALL FOR CLAVE="00000"
```

DELETE FILE

Descripción

Borra un archivo determinado, es importante mencionar que se deberá de incluir su extensión.

Sintaxis

DELETE FILE <nombre archivo>

Ejemplo

```
DELETE FILE BASE.DBF
```

DELETE()

Definición

Regresa T o F si un registro está marcado o no para borrar

Sintaxis

DELETE()

Ejemplo

```
USE DATOS_GR
DO WHILE .NOT. EOF()
 IF RECNO()>='1000'
   DELETE
 ENDIF
ENDDO

DO WHILE .NOT. EOF()
  IF DELETE()
    PACK
  ENDIF
ENDDO
```

DISKSPACE()

Definición

Retorna la cantidad en bytes disponible en la unidad actual.

Sintaxis

DISKSPACE("drive:")

Ejemplo

```
@ 20,1 SAY DISKSPACE("C:")
```

DMY()

Definición

Convierte una expresión fecha dia-mes-año

Sintaxis

DMY(<fecha>)

Ejemplo

```
SET CENTURY ON
@1,80 SAY DMY(DATE())
```

DOW()

Definición

Devuleve el número correspondiente al dia de la semana de la fecha en curso

Sintaxis

DOW(<fecha>)

Ejemplo

```
FECHA=DATE()
DIA=DOW(FECHA)
IF DIA=6 .OR. DIA=7
  @1,1 SAY "DIA DE DESCANSO"
ELSE
```

```
  @1,1 SAY "BUEN DIA DE TRABAJO"
ENDIF
```

DTOC()

Definición

Transforma una expresión de fecha en una expresión de caracteres

Sintaxis

DTOC(<fecha>)

Ejemplo

```
FECHA=DATE()
@1,1 SAY "LA FECHA DE HOY ES :"+DTOC(FECHA)
```

EMPTY()

Definición

Verifica si el contenido de una variable es nulo. Si el resultado es positivo retorna .T., en caso contrario, retorna .F.

Sintaxis

EMPTY(expresión)

Ejemplo

```
IF EMPTY(CVE_NUM) THEN
    @1,1 SAY "DATO VACIO"
ENDIF
```

EOF()

Definición

Busca el fin de archivo

Sintaxis

EOF()

Ejemplo

```
SELECT 1
USE DATOS_GR
DO WHILE .NOT. EOF()
   ? CLAVE,NOMBRE
ENDDO
```

EVALUATE()

Definición

Evalúa una expresión de tipo caracter devolviendo el valor de ésta.

Sintaxis

EVALUATE("expresión")

Ejemplo

```
STORE 0 TO X,Y
OPERACION=SPACE(254)
@1,1 SAY "PRIMER VALOR:" GET X                    // X=7
@2,1 SAY "SEGUNDO VALOR:" GET Y                   // Y=5
@3,1 SAY "TECLEE LA OPERACIÓN: " GET OPERACION
                                                  //OPERACION="X+Y"
READ
RESUL=EVALUATE(OPERACION)
? RESUL                                           // RESUL=35
ENDDO
```

FCOUNT()

Definición

Devuelve el número de campos que existen en un fichero de base de datos activa

Sintaxis

FCOUNT(<base> O <número de select>)

Ejemplo

```
USE DATOS_G
ELEM=FCOUNT()
FOR Y=1 TO ELEM
  @Y,1 SAY NOMBRE
  SKIP
NEXT
```

FIELD()

Definición

Regresa el nombre de cada campo en una base de datos activa, se le deberá de pasar como parámetro un número que representa la posición del campo dentro de la base de datos.

Sintaxis

FIELD(<# posición>)

Ejemplo

```
USE DATOS_G
ELEM=FCOUNT()
FOR Y=1 TO ELEM
  @Y,1 SAY FIELD(Y)
```

```
  SKIP
NEXT
```

FILE()

Definición

Determina si existe un archivo o no

Sintaxis

FILE("archivo")

Ejemplo

```
SELECT 1
USE DATOS_GR
IF FILE ("DATOS_GR.IDX") THEN
  SET INDEX TO DATOS_GR
ELSE
  INDEX ON CLAVE TO DATOS_GR
ENDIF
```

FIND

Definición

Busca un dato llave en un archivo indexado. Para utilizarlo con cadena de caracteres es necesario utilizar el operador macro (&)

Sintaxis

FIND <llave>

Ejemplo

```
SELECT 1
USE DATOS_G INDEX DATOS_G
```

```
NO_EMPLEADO=0
@1,1 SAY "CLAVE : " GET NO_EMPLEADO
READ
FIND NO_EMPLEADO
IF FOUND()
  @1,50 SAY NOMBRE
ENDIF
```

FOUND()

Definición

Retorna el valor lógico .T. si un comando FIND, LOCATE, CONTINUE o SEEK encuentra un registro solicitado por el programa.

Sintaxis

FOUND()

Ejemplo

```
SELECT 1
USE  ALMACEN INDEX ALMACEN
SELECT 2
USE VENTAS INDEX VENTAS
CONTINUA=.T.
DO WHILE CONTINUA
  @ 10,1 SAY "TECLEE LA CLAVE : " GET WCLAVE
  READ
  SELECT 1
  SEEK WCLAVE
  IF FOUND()
    @ 10,25 SAY DESCRIPCION
  ELSE
    @10,25 SAY "CLAVE NO ENCONTRADA"
  ENDIF
ENDDO
RETURN
```

GATHER

Definición

Copia el contenido de los elementos de un arreglo o memoria en un registro de la base de datos activa

Sintaxis

GATHER FROM <arreglo> | <variable de memoria>
[FIELDS <lista de campos>] [memo]

Ejemplo

```
USE DATOS_G
SCATTER MEMORIA
@18,3 SAY "NO EMPLEADO: " GET NO_EMPLEADO
READ
IF LASTKEY() <>27
  GATHER MEMORIA
ENDIF
RETURN
```

GETDIR()

Descripción

Muestra una ventana con el contenido del directorio o ruta específica indicada

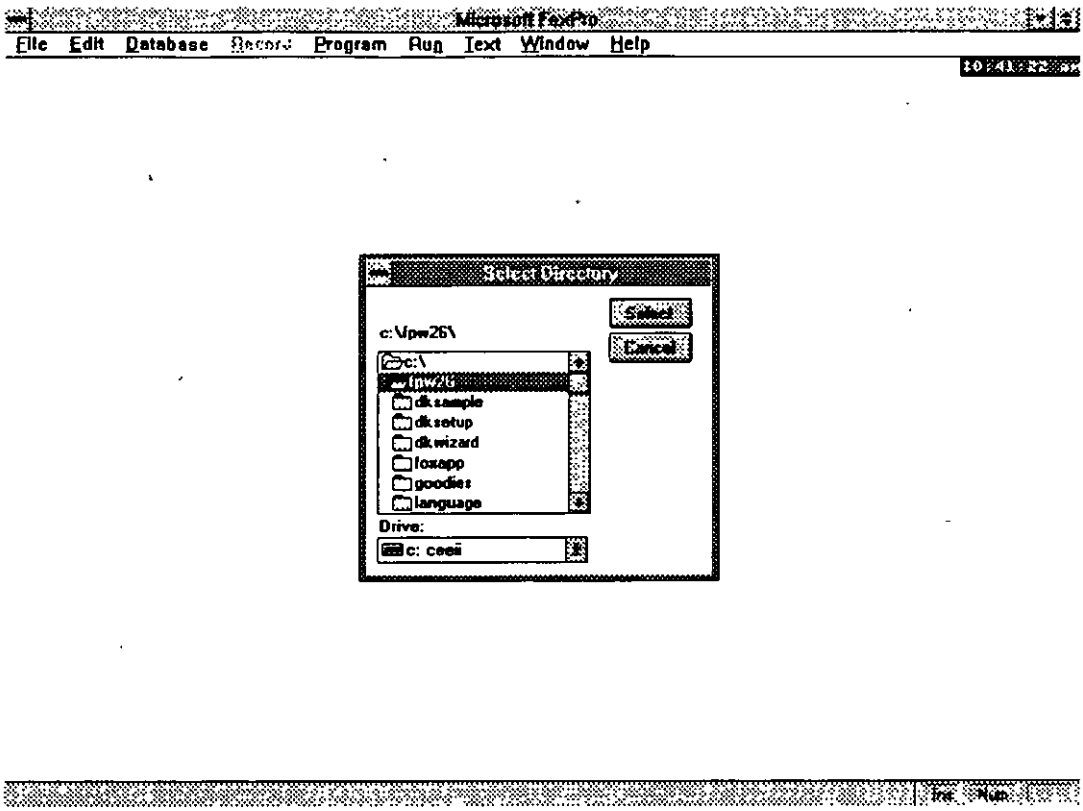
Sintaxis

GETDIR("<ruta>")

Ejemplo

```
CLEAR
@5,5
?GETDIR("C:\FOXPROW)
```

Esto nos mostraría la siguiente pantalla



IIF()

Definición

Comprende una condición a cumplir y devuelve siempre uno de los dos resultados posibles, es un condicional sólo con dos soluciones posibles.

Sintaxis

IIF(<condición>,<exp. verdadera>,<expresión falsa>)

Ejemplo

```
USE DATOS_G
@1,1 SAY "VIGENTE : " GET VIGE VALID "SN"
READ
IIF(VIGE="S",LETRERO="VIGENTE",LETRERO="DE BAJA")
@1,50 SAY LETRERO
```

INKEY()

Definición

Devuelve el valor numérico de la tecla pulsada.
Detiene el proceso hasta que se presiona una tecla o ha recorrido un cierto intervalo de tiempo y retorna un número que representa la tecla presionada mas recientemente.

Sintaxis

INKEY(segundos)

Ejemplo

```
USE DATOS_GR
@ 1,1 SAY "DESPLIEGUE DE INFORMACIÓN"
INKEY(0)
DO WHILE !EOF()
  ?CLAVE,NOMBRE
  SKIP
ENDDO
```

INT()

Definición

Retorna la parte entera de la expresión numérica suministrada como argumento. La función INT() no redondea el número,

simplemente no toma en cuenta los decimales. Semejante a la función ROUND()

Sintaxis

INT(#)

Ejemplo

```
@1,1 SAY "ELEMENTO 1:" GET A
@2,1 SAY "ELEMENTO 2: " GET B
READ
SUMA=A/B
ENTERO=INT(SUMA)
```

ISPRINTER()

Definición

Retorna .T. si la impresora en el puerto paralelo se encuentra conectada y lista para usarse

Sintaxis

ISPRINTER()

Ejemplo

```
IF ISPRINTER()
  SET DEVICE TO PRINT
ELSE
 @ 1,1 SAY "IMPRESORA NO EXISTENTE O FUERA DE LINEA"
ENDIF
```

KEY()

Definición

Devuelve el campo correspondiente a un archivo índice.
Por el orden en que fue abierto un archivo índice se puede averiguar el campo al que corresponde.

Sintaxis

KEY ([<archivo índice>],[<área>])

Ejemplo

```
USE DATOS_G
INDEX ON NO_EMPLE TAG EMPLEADO OF DATOS_G
INDEX ON NOMBRE TAG NOMBRE OF DATOS_G
? KEY(1)    // NO_EMPLE
? KEY(2)    // NOMBRE
```

LASTKEY()

Definición

Retorna el valor ASCII de la última tecla presionada.

Sintaxis

LASTKEY()

Ejemplo

```
IF  LASTKEY() = 27
  RETURN
ENDIF
```

LEFT()

Definición

Sustrae un substring a partir de la izquierda, iniciando con el primer caracter y tomándolo de longitud igual al especificado por el argumento de la función.

Sintaxis

LEFT(string,cuantos)

Ejemplo

```
CADENA="EL NOMBRE ES "
SUB_CADENA=LEFT(CADENA,5)
```

LEN()

Definición

Determina la longitud de un string

Sintaxis

LEN(string)

Ejemplo

```
IF LEN(CADENA)<0
   @1,1 "LA VARIABLE ESTA VACIA"
ENDIF
```

LOCATE

Definición

Localiza información en una base de datos no indexada

Sintaxis

LOCATE <scope> FOR <condición>

Ejemplo

```
SELECT 1
USE  ALMACEN
SELECT 2
USE VENTAS INDEX VENTAS
CONTINUA=.T.
DO WHILE CONTINUA
  @ 10,1 SAY "TECLEE LA CLAVE : " GET WCLAVE
  READ
  SELECT 1
  DO WHILE .NOT. EOF()
     LOCATE FOR CLAVE=WCLAVE
     @ 10,25 SAY DESCRIPCION
     CONTINUE
  ENDDO
ENDDO
RETURN
```

LOCFILE()

Definición

Permite localizar un archivo en un disco

Sintaxis

LOCFILE(<exp caracter>)

Ejemplo

? LOCFILE('DATOS.DBF') / Si lo encuentra imprimirá la ruta

LTRIM()

Definición

Elimina los espacios en blanco a la izquierda de una cadena de caracteres

Sintaxis

LTRIM(string)

Ejemplo

```
CADENA=SPACE(20)
@1,2 SAY "TECLEE LA CADENA" GET CADENA
READ
CADENA=LTRIM(CADENA)
```

LUPDATE()

Definición

Retorna la fecha de la última actualización de la base de datos en uso

Sintaxis

LUPDATE()

Ejemplo

```
IF LUPDATE() <>DATE()
  @1,1 SAY "DEBE DE RESPALDAR"
ENDIF
```

NDX()

Definición

Regresa el nombre del archivo índice (*.IDX) activo.

Sintaxis

NDX(<# del índice>)

Ejemplo

```
USE DATOS_G INDEX DATOS_G
?NDX(1)                  // C:\FPW26\DATOS_D.IDX
```

ON ERROR

Definición

Si se produce algún tipo de error FoxPro realizará la instrucción descrita después de la instrucción ON ERROR. Para el caso del manejo de errores el sistema enviará dos parámetros cuando existe algún error, ERROR(), MESSAGE(). En el primero almacena el número del error y en el segundo el mensaje que envía el sistema.

Sintaxis

ON ERROR <comando>
ON ERROR DO <función> WITH ERROR(),MESSAGE()

Ejemplo

```
USE DATOS_E
INDEX ON DATOS_E
ON ERROR EXIT
```

```
USE DATOS_E
INDEX ON DATOS_E
ON ERROR DO ERR_FUN
.
.
RETURN

FUNCTION ERR_FUN
WAIT "ERROR AL ABRIR LA BASE DE DATOS" WINDOW
RETURN


USE DATOS_E
INDEX ON DATOS_E
ON ERROR EXIT
```

```
USE DATOS_E
INDEX ON DATOS_E
ON ERROR DO ERROR_MEN WITH ERROR(),MESSAGE()
.
.
RETURN

FUNCTION ERR_MEN
PARAMETERS NO_ERROR,MENSAJE
@ 15,35 TO 18,75
@ 16,36 SAY "ERROR NO."+CHR(NO_ERROR)

DO CASE
CASE NO_ERROR=1230
        MENSAJE ="DEMASIADOS ARGUMENTOS"
CASE NO_ERROR=3
        MENSAJE ="BASE DE DATOS ABIERTA"
.
.
ENDCASE
@ 17,36 SAY MENSAJE
RETURN
```

ON ESCAPE

Definición

Si se oprime la tecla ESC se ejecutará la orden indicada. La tecla ESC provoca que el programa sea interrumpido

Sintaxis

ON ESCAPE <instrucción>

Ejemplo

```
ON ESCAPE DO PROG_ESC
.
.
RETURN

FUNCTION PROG_ESC
WAIT "TERMINA EL PROGRAMA " WINDOW
QUIT
```

ON KEY

Definición

En este caso cuando la tecla que se definió se oprime, el programa se interrumpe para realizarse la órden descrita. Si ON KEY no tiene nombre de tecla, se ejecutará el comando descrito oprimíendose cualquier que no sea ESC.

Sintaxis

ON KEY [<#>][<instrucción>]

Ejemplo

```
ON KEY =28 DO PROGRAMA
.
RETURN

FUNCTION PROGRAMA
?INSTRUCCIONES A REALIZARSE
```

ON KEY LABEL

Definición

En este caso la instrucción se ejecutará cuando la tecla con un nombre (o etiqueta) determinado(a) sea oprima. Este comando enviará a la rutina de manejo de la tecla una variable que corresponderá a la variable que se está leyendo en ese momento o el nombre del campo de la base de datos donde se encuentra el apuntador.

Sintaxis

ON KEY LABEL <etiqueta de la tecla>
ON KEY LABEL <etiq. de la tecla> DO <nom. proc.> WITH VARREAD()

donde:

VARREAD() es la función que devuelve el nombre del campo

Ejemplo

```
ON KEY LABEL F1 DO AYUDA
.
.
RETURN

FUNCTION AYUDA
?PROGRAMA DE AYUDA
RETURN
```

De otra forma:

```
BROWSE
ON KEY LABEL F1 DO LEE WITH VARREAD()
.
.

FUNCTION LEE
PARAMETERS CAMPO
DO CASE
   CASE  CAMPO="NUM_EMPLEADO"
         ?DESPLIEGA AYUDA DEL NÚMERO DEL EMPLEADO
   CASE CAMPO="CATEGORIA"
         ?DESPLIEGA AYUDA DE LA CATEGORÍA
   OTHERWISE
         ?NO EXISTE AYUDA EMPLEADO
ENDCASE
```

PACK

Definición

Borra los registros marcados para borrar

Sintaxis

PACK

Ejemplo

```
USE DATOS_GR
DO WHILE RECNO()>100
  DELETE
  SKIP
ENDDO
PACK
```

PADC, PADR, PADL

Definición

Alinea un letrero en una longitud determinada y rellena los espacios con el caracter que se le indica

Sintaxis

Izquierda	PADL("<cadena>",longitud,"caractr. de relleno")
Derecha	PADR("<cadena>",longitud,"caractr. de relleno")
Centro	PADC("<cadena>",longitud,"caractr. de relleno")

Ejemplo

```
LETRERO="TITULO DEL TEXTO"
PAD(LETRERO,80," ")
```

PROPER()

Definición

Devuelve la cadena de caracteres cambiando la primera letra a mayúscula.

Sintaxis

PROPER(<exp caracter>)

Ejemplo

```
?PROPER("foxpro")          // Foxpro
```

RECALL

Definición

Quita las marcas de borrado en los registros

Sintaxis

RECALL

Ejemplo

```
USE DATOS_GR
DO WHILE RECNO()>100
  DELETE
  SKIP
ENDDO
RECALL
```

RECCOUNT()

Definición

Retorna el número total de registros del archivo en uso, incluídos los marcados para eliminación.

Sintaxis

RECCOUNT()

Ejemplo

```
@1,1 "EL NUMERO TOTAL DE REGISTRO ES " + STR(RECCOUNT())
```

RECNO()

Definición

Devuelve el número del registro donde se encuentra el apuntador de la base de datos activa.

Sintaxis

RECNO()

Ejemplo

```
USE DATOS_G
SCAN
  @1,1 SAY RECNO()
  @1,15  SAY NOMBRE
ENDSCAN
```

REINDEX

Definición

Reconstruye o regenera el archivo índice activo. También se utiliza para compactar un archivo índice de extensión *.IDX.

Sintaxis

REINDEX

Ejemplo

```
USE DATOS_GR INDEX DATOS_GR
@1,1 GET WCLAVE
@2,1 GET WNOMBRE
READ
APPEND BLANK
REPLACE  CLAVE     WITH WCLAVE
REPLACE  NOMBRE   WITH WNOMBRE
REINDEX
```

RELEASE

Definición

Borra la memoria de las variables, menús, popups, opciones y ventanas

Sintaxis

RELEASE <objeto a borrar> <nombre>

Ejemplo

```
DEFINE WINDOW VENTA FROM 1,1 TO 5,5
ACTIVATE WINDOW VENTA
@1,1 SAY "NUM. EMPLEADO: " GET NUM_EMPLEDO
READ
APPEND BLANK
REPLACE NUM_EMPLE  WITH  NUM_EMPLEADO
DEACTIVATE WINDOW VENTA
RELEASE WINDOW VENTA
RETURN
```

—

REPLACE

Definición

Reemplaza los campos de un registro de la base de datos activa sustituyéndolos por la expresión indicada en cada caso

Sintaxis

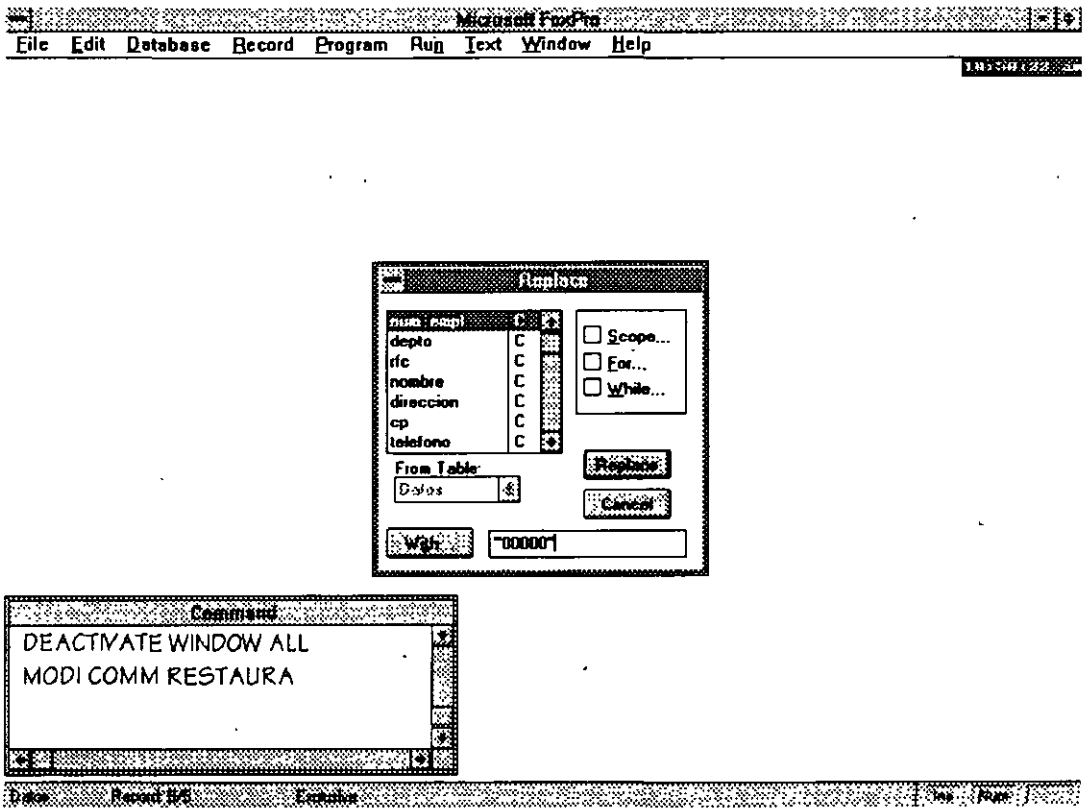
REPLACE <campo base de datos> WITH <variable I valor>

Ejemplo

```
DEFINE WINDOW VENTA FROM 1,1 TO 5,5
ACTIVATE WINDOW VENTA
@1,1 SAY "NUM. EMPLEADO: " GET NUM_EMPLEDO
```

```
READ
APPEND BLANK
REPLACE NUM_EMPLE  WITH  NUM_EMPLEADO
DEACTIVATE WINDOW VENTA
RETURN
```

Por medio de menú se podría manipular la instrucción Replace para el caso de querer reemplazar un campo con algún caracter o número en especial, pudiéndole determinar condiciones de reemplazo y número de elementos, esto lo vemos claramente en la pantalla presentada al llamar al menú Record-replace



REPLICATE()

Sintaxis

REPLICATE(string, número)

Definición

Repite un número determinado de veces un string

Ejemplo

```
@1,1 SAY REPLICATE("-",50)
@2,1 SAY TITULO
```

RESTORE FROM

Definición

Recupera de disco, las variables de memoria grabadas por la instrucción SAVE TO.

Sintaxis

RESTORE FROM <var de memoria>
RESTORE FROM MEMO <campo memor>

Ejemplo

```
RESTORE FROM EMPLEADO          // EMPLEADO.MEM
USE DATOS_G
NUM_E=VAL(NUM_EMPLEADO)+1
NOMBRE1=SPACE(30)
DO WHILE .T.
  @1,01 SAY "NUM. EMPLEADO:"
  @1,10 SAY NUM_E
  @2,1 SAY "NOMBRE: " GET NOMBRE1
  READ

  APPEND BLANK
  REPLACE NUM_EMPL WITH STR(NUM_E)
  NUM_E=NUM_E+1
ENDDO
NUM_EMPLEADO=STR(NUM_E)
SAVE TO EMPLEADO
```

RESTORE SCREEN

Definición

Recupera una pantalla salvada en una variable de memoria con SAVE SCREEN.

Sintaxis

RESTORE SCREEN FROM <nombre_pantalla>

Ejemplo:

```
CLEAR
DEFINE WINDOW PANTA1 FROM 1,1 TO 15,70
ACTIVATE WINDOW PANTA1
WCLAVE=SPACE(5)
WNOMBRE=SPACE(30)
WPUESTO=SPACE(5)
WSUELDO=0
@1,1 GET WCLAVE
@2,1 GET WNOMBRE
READ
SAVE SCREEN TO PANTALLA
CLEAR
DEFINE WINDOW PANTA2 FROM 5,5 TO 21,65
ACTIVATE WINDOW PANTA2
@6,6 GET WPUESTO
@7,6 GET WSUELDO
DEACTIVATE WINDOW PANTA2
RESTORE SCREEN FROM PANTALLA
ACTIVATE WINDOW PANTA1
```

RESTORE WINDOW

Definición

Restaura una ventana guardada en una variable de memoria

Sintaxis

RESTORE WINDOW <nombre ventana> FROM <variable de memoria>

Ejemplo

```
CLEAR
DEFINE WINDOW PANTA1 FROM 1,1 TO 15,70
ACTIVATE WINDOW PANTA1
WCLAVE=SPACE(5)
WNOMBRE=SPACE(30)
WPUESTO=SPACE(5)
WSUELDO=0
@1,1 GET WCLAVE
@2,1 GET WNOMBRE
READ
SAVE WINDOW PANTA1  TO PANTALLA
CLEAR
DEFINE WINDOW PANTA2 FROM 5,5 TO 21,65
ACTIVATE WINDOW PANTA2
@ 6,6 GET WPUESTO
@7,6 GET WSUELDO
READ
DEACTIVATE WINDOW PANTA2
RESTORE WINDOW PANTA1  FROM PANTALLA
ACTIVATE WINDOW PANTA1
```

RETURN

Definición

Regresa el control a la instrucción siguiente

Sintaxis

RETURN

Ejemplo

```
USE DATOS_GR INDEX DATOS_GR
@1,1 GET WCLAVE
@2,1 GET WNOMBRE
READ
```

```
APPEND BLANK
REPLACE  CLAVE     WITH WCLAVE
REPLACE  NOMBRE    WITH WNOMBRE
RETURN
```

RIGHT()

Definición

Extrae un determinado número de caracteres a partir de la derecha

Sintaxis

RIGHT(string,cuantos)

Ejemplo

```
CADENA="EL NOMBRE ES "
SUB_CADENA=RIGHT(CADENA,5)
```

ROW()

Definición

Devuelve el renglón donde se encuentra actualmente el cursor en la pantalla

Sintaxis

ROW()

Ejemplo

```
USE DATOS_E
SCAN
  @ROW()+1,15 SAY NOMBRE
ENDSCAN
RETURN
```

RTRIM()

Definición

Elimina los blancos a la derecha de una expresión de caracteres.

Sintaxis

RTRIM(string)

Ejemplo

```
NOMBRE=SPACE(20)
@1,1 SAY "NOMBRE: " GET NOMBRE
READ
CADENA= RTRIM(NOMBRE)
```

SAVE SCREEN

Definición

Salva una pantalla en memoria

Sintaxis

SAVE SCREEN TO <pantalla> .

Ejemplo

```
CLEAR
@1,1 TO 15,70
@1,1 GET WCLAVE
@2,1 GET WNOMBRE
READ
SAVE SCREEN TO PANTALLA
CLEAR
@ 5,5 TO 20,78
@ 6,6 GET  WPUESTO
@ 7,6 GET  WSUELDO
RESTORE SCREEN FROM PANTALLA
```

SAVE TO

Definición

Salva las variables de memoria en un archivo de memoria. La extensión de no especificarse le dará al archivo la de mem.

Sintaxis

SAVE TO <nombre archivo memo> [all | like | except <máscara>]

Ejemplo

```
PRIVATE VALOR=12
SAVE TO MEMORIA
VALOR1=15
VALOR2=25
SAVE ALL TO MEMO2
```

SCATTER

Definición

Copia los elementos de una base de datos en un arreglo o en una variable de memoria.

Sintaxis

SCATTER [FIELDS <campos>] [MEMO] TO <arreglo>

Ejemplo

```
USE DATOS_E
SCATTER FIELDS NOMBRE,RFC,TELEFONO TO ARREGLO
DEFINE WINDOW VENTA FROM 1,5 TO 15,45
ACTIVATE WINDOW VENTA
@ 2,1 SAY "NOMBRE: " GET ARREGLO(1)
@ 3,1 SAY "RFC: "  GET ARREGLO(2)
@ 4,1 SAY "TELEFONO :  " GET ARREGLO(3)
READ
```

SCROLL

Definición

Permite mover una ventana una o varios renglones hacia arriba o hacia abajo.

Sintaxis

SCROLL (ren1,col1,ren2,col2,#líneas)

Ejemplo

```
REN=6
COND=.T.
DO WHILE COND
 STORE SPACE(30) TO NOMBRE,DIRECCION,COLONIA
 @ REN,COL()+2 GET NOMB PICTURE "@!"
 @ REN,COL()+2 GET DIRE PICTURE "@!"
 @ REN,COL()+2 GET COLO PICTURE "@!"
 READ
 COND=(.NOT. EMPTY(NOMB))
 IF REN>=17
  SCROLL(5,2,17,65,1)
  LOOP
 ENDIF
 REN=REN+1
ENDDO
```

SEEK

Definición

Busca un valor en un archivo indexado

Sintaxis

SEEK <variable>

Ejemplo

```
SELECT 1
USE  ALMACEN INDEX ALMACEN
SELECT 2
USE VENTAS INDEX VENTAS
CONTINUA=.T.
DO WHILE CONTINUA
    @ 10,1 SAY "TECLEE LA CLAVE : " GET WCLAVE
    READ
    SELECT 1
    SEEK WCLAVE
    @ 10,25 SAY DESCRIPCION
ENDDO
RETURN
```

SET FILTER TO

Definición

Filtra un número determinado de registros que cumplen con una condición determinada.

Sintaxis

SET FILTER TO <condición>

Ejemplo

```
USE DATOS_G
SET FILTER TO NUM_DEPTO="5000"
COPY TO LISTA5
SET FILTER TO
```

SKIP

Definición

Salta al siguiente registro

Sintaxis

SKIP

Ejemplo

```
SELECT 1
DO WHILE !EOF()
   ? CLAVE,NOMBRE
   SKIP
ENDDO
```

SORT TO

Definición

Sortea una base de datos sobre un campo determinado dejando el resultado en otra base de datos

Sintaxis

SORT TO <nuevo archivo> ON <campo> [/A, /D, /C]
[<campo2> [/A, /D, /C]]
[FOR]
[WHILE]
[FIELDS]

/A Ascendente
/D Descendente
/C Ignora el case

Ejemplo

```
SELECT 1
USE DATOS_GR
SORT TO DATOS_DOR ON CLAVE FOR CLAVE>"1000" /A
```

STR()

Definición

Convierte una variable de tipo numérico a una de tipo caracter.

Sintaxis

STR(número,tamaño,decimales)

Ejemplo

```
@1,10 SAY "TECLEE EL NUMERO: " GET NUMERO
READ
@2,20 SAY "EL NUMERO TECLEADO: "+STR(NUMERO,5,2)
```

SUBSTR()

Definición

Sustrae una cadena de caracteres de otra

Sintaxis

SUBSTR(cadena,pos.inic,cuantos)

Ejemplo

```
CADENA="EL NOMBRE ES "
SUB_CADENA=RIGHT(CADENA,1,5)
```

SUM

Definición

Obtiene la suma de los diferentes valores que toma un campo numérico.

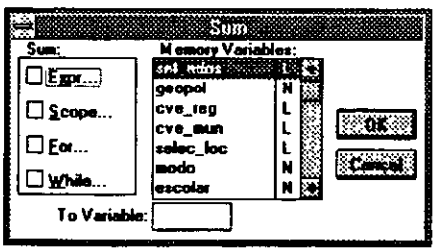
Sintaxis

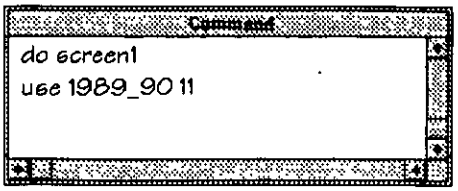
SUM <lista de campos> TO <lista de variables> [FOR | WHILE <condición]

Ejemplo

```
USE NOMINA
SUM SUELDO, ISR,IVA TO SUMA1,SUMA2,SUMA3
?SUMA1,SUMA2,SUMA3
```

Este mismo proceso es posible realizarlo por medio del menú, llamando a la opción Database-SUM, teniéndose la siguiente pantalla:





Podemos observar que de igual manera nos permitirá definirle condiciones sobre el campo a sumar y en que variable se almacenará el resultado.

TIME()

Definición

Hora del sistema

Sintaxis

TIME()

Ejemplo

```
@ 1,1 SAY "LA HORA DEL SISTEMA ES "
@2,1 SAY TIME()
```

TOTAL

Definición

Obtiene el total en los campos numéricos de una base de datos.

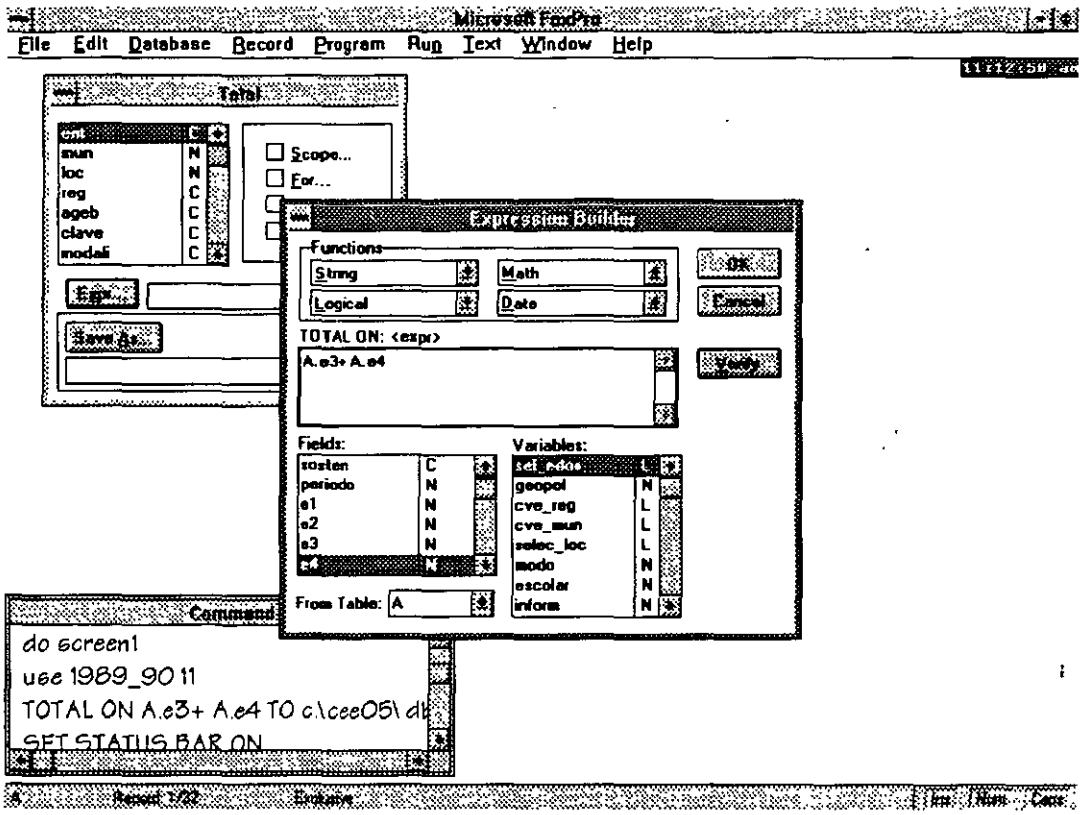
Sintaxis

TOTAL TO <archivo.dbf> ON <llave> ON <exp. de campos o combinación>
FIELDS<lista de campos> [FOR | WHILE]

Ejemplo

```
USE NOMINA INDEX NUM_EMPLE
TOTAL ON NUM_EMPLE FIELDS SUELDO,DESCUENTOS TO SUEL,DESC
```

Esto mismo lo podemos manejar por medio del menú, Database-Total y se nos mostrará la siguiente pantalla:



Aquí nos permitirá crear la expresión determinar campos y condiciones.

TRANSFORM

Definición

Transforma una expresión según formato especificado

Sintaxis

TRANSFORM(<exp>,<formato>)

Ejemplo

```
TRANSFORM("nombre","!@")  // NOMBRE
TRANSFORM(58790.89,"$999,999.99")  // $58,790.89
```

TYPE()

Definición

Determina el tipo de dato del que se envía el argumento

Sintaxis

TYPE(string)

C CARACTER
D FECHA
N NUMÉRICO
M MEMO
L LÓGICO
U INDEFINIDO

Ejemplo

```
IF TYPE(CVE_USUARIO)="C"
  @1,1 SAY "LA CLAVE ES DE TIPO CARACTER"
ENDIF
```

UPPER()

Definición

Cambia a mayúsculas la cadena

Sintaxis

UPPER(string)

Ejemplo

```
CADENA="el letrero en minúsculas"
CADENA=UPPER(CADENA)
```

VAL()

Definición

Convierte de caracter a numérico

Sintaxis

VAL(string)

Ejemplo

```
CLAVE="00090"
NUMERO_CH=VAL(CLAVE)
CLAVE=STR(NUMERO_CH)
```

VARREAD()

Definición

Devuelve el nombre del campo u objeto que se encuentra activado después de un READ.

Sintaxis

VARREAD()

Ejemplo

```
SET KEY = -1 TO NOMBRE          // F2 llama al proc. nombre
WNOMBRE=SPACE(30)
WCATEGO=SPACE(5)
@1,1 SAY "NOMBRE : " GET WNOMBRE
```

```
@2,1 SAY "CATEGORIA:" GET WCATEGO
@16,12 SAY "[F2] AYUDA"
READ


PROCEDURE NOMBRE
DO CASE
    CASE VARREAD="WNOMBRE"
         @20,20 SAY "EL NOMBRE NO PUEDE QUEDAR EN BLANCO"
   CASE VARREAD="WCATEGO"
         ?DESPLEGAR CATÁLOGO DE CATEGORÍAS
   OTHERWISE
         @20,20 SAY "VARIABLE SIN AYUDA"
ENDCASE
RETURN
```

WAIT

Definición

Espera que se oprima una tecla para continuar, pudiendo desplegar un letrero, éste deberá de definirse entre comillas, en el caso de que sólo se desee que se espere la siguiente tecla se teclearán las comillas sin contenido. Para el caso de utilizar la opción de WINDOW, le desplegará al usuario el mensaje en una ventana que aparecerá en la parte superior derecho de la pantalla.

Sintaxis

WAIT [<expresión>] [WINDOW | TIMEOUT | CLEAR]

Ejemplo

```
SELECT 1
USE DATOS_GR
SORT  ON CLAVE TO DATOS_SOR
WAIT "TERMINO DE SORTEAR POR CLAVE" WINDOW
```

5.1 Ejercicio

Utlizando los comando antes vistos realizar el siguiente ejercicio

1. Crear una pantalla principal donde despliegue en la parte inferior fecha, hora, espacio en disco y unidad de disco de trabajo.

2. Crear una pantalla de captura para nuevos puestos, revisando si existe clave repetida, avisarle al usuario y mostrarle en pantalla la clave existente y su descripción

3. Crear la pantalla de captura para la nómina desplegándose en caso de oprimir la tecla de ayuda el catálogo de empleados y mostrando en pantalla la definición de cada clave

4. Definir en papel procesos que se desearían hacer con el sistema

5. Crear el módulo de consulta para los datos generales del empleado, donde se nos pregunte el número o nombre del empleado y se nos depligue en pantalla su información

6. Manejo de objetos

6.1 Definición

Definiremos objetos a características propias del paquete como los son: acciones de lectura y escritura de datos, éstos ayudarán a seleccionar la forma de actuar dependiendo del proceso asociado.

Dentro de Foxpro podremos manejar los siguientes objetos:

- Radio buttons
- Push buttons
- Controles Popup
- Check boxes
- Scrolling list
- Spinners

Se definirá y ejemplificará cada uno de los objetos en este capítulo.

En primera instancia veremos las semejanzas que tendrán los objetos entre sí a lo cuál llamaremos generalidades que se definen a continuación:

6.2 Generalidades

- Inicialización

  Llamaremos inicialización al valor con que definimos a la variable que alojará el valor seleccionado en cualquiera de los objetos, pudiendo ser valores carácter o numérico.
  Si inicialmente definimos a la variable como carácter el valor asignado a ésta será el nombre de la opción seleccionada, si la variable fue numérica, entonces el valor será el número de la opción elegida.

Ejemplo:

```
opcion=1
@1,1 get opcion picture "@*R Guardar;Imprimir;Salir"
read
```

Si selecciónо Guardar opción valdrá 1, en el caso de Imprimir opción será 2 y 3 si fue Salir.

```
opcion=" "
@1,1 get opcion picture "@*R Guardar;Imprimir;Salir"
read
```

Si selecciónо Guardàr opción "Guardar", en el caso de Imprimir opción será "Imprimir" y "Salir" si fue Salir.

Es muy importante tomar esto en cuenta para el manejo del resultado en la selección de un valor dentro de los objetos.
Para el caso de listas obtenidas de vectores si la inicialización fue numérica, la variable guardará el valor de la posición del elemento seleccionado y si fue caracter el nombre.

- Opciones por default

Las opciones por default son aquellas que el objeto tomará como el primer valor que tendrá el objeto. Por ejemplo, si tuviéramos lo siguiente:

```
@1,1 get opcion picture "@*R Guardar;Imprimir;Salir" default 2
read
```

El valor definido por el usuario como default es 2, esto es al desplegarse los radio botones tendrán como señalada la opción de Imprimir.

- Despliegue

El despliegue será la forma en que los objetos aparecerán en la pantalla, pudiendo aparecer en forma vertical u horizontal. Esto se utilizará sólo con los objeto radio botones, check boxes, push buttons.
Al objeto le indicaremos la orientación del despliegue mediante las letras H y V, las cuales representan Horizontal y Vertical respectivamente y se anexarán al objeto en la definición del mismo.

Ejemplo

```
@1,1 get opcion picture "@*RH Guardar;Imprimir;Salir" default 2
read
```

En este caso los radio botones será desplegados en forma horizontal.

- Ejecución

La ejecución la definiremos como el momento en que se ejecuta la acción selecionada. Existen dos formas que se lleven a cabo o se asigne el valor a la variable, la primera, en el momento que ha sido seleccionada y la segunda hasta que espere a que se defina todas la opciones y se determine por proceso que ha sido terminada la lectura. Para ello tendremos las opciones N para cuando no queremos que termine al ser seleccionada y T cuando se terminará en el momento de la selección.

Ejemplo

```
@1,1 get opcion picture "@*RHT Guardar;Imprimir;Salir" default 2
  read
```

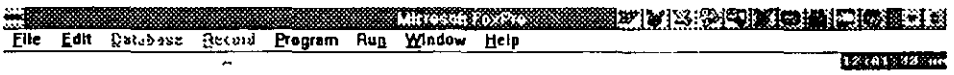
En este caso en el instante en que es seleccionada la opción termina la lectura del objeto y continua con la siguiente instrucción.

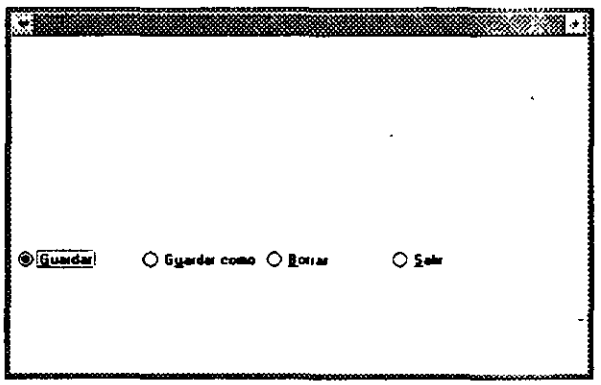
- Teclas calientes

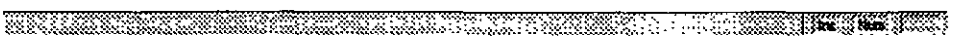
Llamaremos teclas calientes (Hot keys) a aquellas que con tan solo oprimirlas nos seleccionará la opción deseada. Estas son muy utilizadas en el ambiente Windows, en donde las identificaremos por estar o mas iluminadas o por estar subrayadas. Esto es, si queremos llamar a la opción Archivo bastará con oprimir A y desplegará el menú de archivo. Resulta útil este tipo de manipulación ya que se le presenta al usuario otra forma de accesar a las opciones.
Definiremos a las teclas calientes anteponiéndole a la letra una diagonal y un menor que (\<).

Nota: Se deberá tener la precaución de no utilizar la misma tecla como tecla caliente de mas de una opción.

Ejemplo
@1,1 get opcion;
picture "@*RH \<Guardar;G\<uardar como:\<Borrar;\<Salir";default 2
read







6.3 Radio botones (Radio buttons)

6.3.1 Definición

Los radio botones son objetos conocidos, manejados en diversas aplicaciones en Windows. Los radio botones representan un conjunto de opciones a elegir, por cada uno de ellos tendremos un círculo vacío asociado, él cuál marcaremos con la barra espaciadora o con el mouse para indicarle al sistema que esa ha sido nuestra elección, dentro del círculo aparecerá un nuevo círculo concéntrico relleno.
Dentro de éste tipo de objetos sólo será posible elegir una de las opciones.

6.3.2 Sintaxis

La sintaxis se referirá en este caso al tipo de la llamada al objeto dentro de un GET, pudiendo ser mediante el PICTURE o por FUNCTION

PICTURE

```
@col,ren SAY "texto" GET opcion;
        PICTURE "@*R Guardar;Borrar;Copiar;Salir" default =1
```

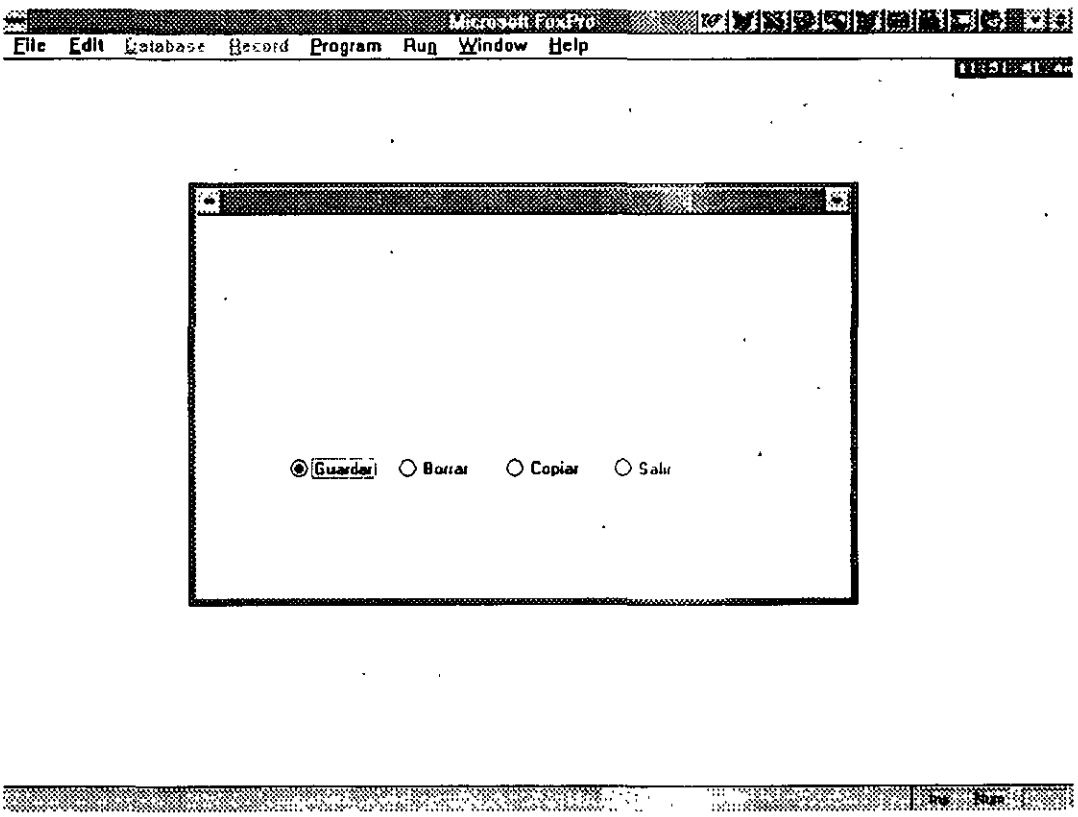
FUNCTION

```
@col,ren SAY "texto" GET opcion;
        FUNCTION "*R Guardar;Borrar;Copiar;Salir" default =1
```

Como se puede observar en la sintaxis lo que definirá el tipo de objeto a utilizarse será el * y la R mayúscula. Las características a definirle serán: el texto de cada botón (opciones), la separación o distancia entre ellos, la forma de despliegue y la terminación en la lectura, tipo de letra y tamaño (que se vió en generalidades)

Ejemplo:

```
@ 13.923,14.400 GET opcion ;
    PICTURE "@*RHN Guardar;Borrar;Copiar;Salir" ;
    SIZE 1.308,11.667,1.833 ;
    DEFAULT 1 ;
    FONT "MS Sans Serif", 8 ;
    STYLE "BT"
```



Como se platicó en generalidades dependiendo de la inicialización será el valor que se alojará en la variable de lectura.

6.4 Botones para oprimir (Push buttons)

6.4.1 Definición

Este tipo de objetos nos permite crear botones en la pantalla, los cuáles al oprimirse se ejecutará algún proceso.
Estos botones pueden contener un texto determinado o imágenes.
El cursor se deslizará de botón en botón dando la apariencia de sobresaliente y oprimiendo <enter> o señalándolo directamente con el mouse la opción se elegirá.

6.4.2 Sintaxis

Tendremos como sintaxis nuevamente dentro del PICTURE o o de FUNCTION en el GET.
@ren,col GET <variable> FUNCTION "*B <lista de opciones>" |
PICTURE "@*B <lista de opciones>";
DEFAULT #;
SIZE <# altura>,<# ancho>,<# distancia>;
MESSAGE <mensaje>;
VALID <expresión que se ejecutará cuando la opción valida una condición>;
WHEN <expresión que se ejecutará bajo alguna condición>;
COLOR SCHEME <lista de colores>

Para el caso de trabajar con imágenes tendremos la siguiente sintaxis:

@ren,col GET <variable> FUNCTION "*B <lista de opciones>" |
PICTURE "@*B <lista de opciones";
DEFAULT #;
SIZE <# altura>,<# ancho>,<# distancia>;
MESSAGE <mensaje>;
VALID <expresión que se ejecutará cuando la opción valida una condición>;
WHEN <expresión que se ejecutará bajo alguna condición>;
COLOR SCHEME <lista de colores>

en lista de opciones se tendrá :

```
(LOCFILE("disco.ico","BMP|ICO|PCT|ICN","Where is disco?"))+";"+;
(LOCFILE("grafica.ico","BMP|ICO|PCT|ICN","Where is grafica?"))+";" + ;
(LOCFILE("impreso.ico","BMP|ICO|PCT|ICN","Where is impreso?"))+";"+;
(LOCFILE("salida.ico","BMP|ICO|PCT|ICN","Where is salida?")) ;
```

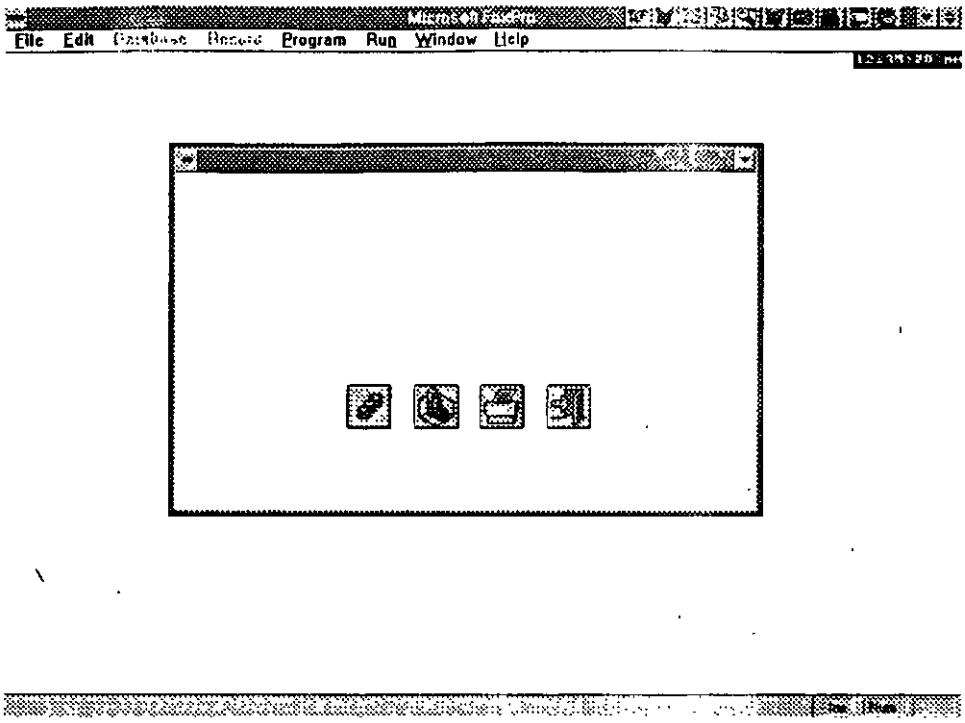
donde la función LOCFILE localizará el archivo definido en las primeras comillas, el tipo de archivo mostrado en las segundas comillas y como tercer parámetro la pregunta de donde está el archivo, se concatenarán las opciones mediante signos de mas.

Esto se verá claramente en el siguiente ejemplo:

Ejemplo:

```
@ 13.846,28.800 GET opcion ;
   PICTURE "@*BHT " + ;
  (LOCFILE("disco.ico","BMP|ICO|PCT|ICN","Where is disco?")) + ";" + ;
  (LOCFILE("grafica.ico","BMP|ICO|PCT|ICN","Where is grafica?"))+ ";" + ;
  (LOCFILE("impreso.ico","BMP|ICO|PCT|ICN","Where is impreso?"))+";"+ ;
  (LOCFILE("salida.ico","BMP|ICO|PCT|ICN","Where is salida?")) ;
  SIZE 2.923,7.600,3.800 ;
  DEFAULT 1 ;
  FONT "MS Sans Serif", 8
```

Quedándonos la siguiente pantalla:

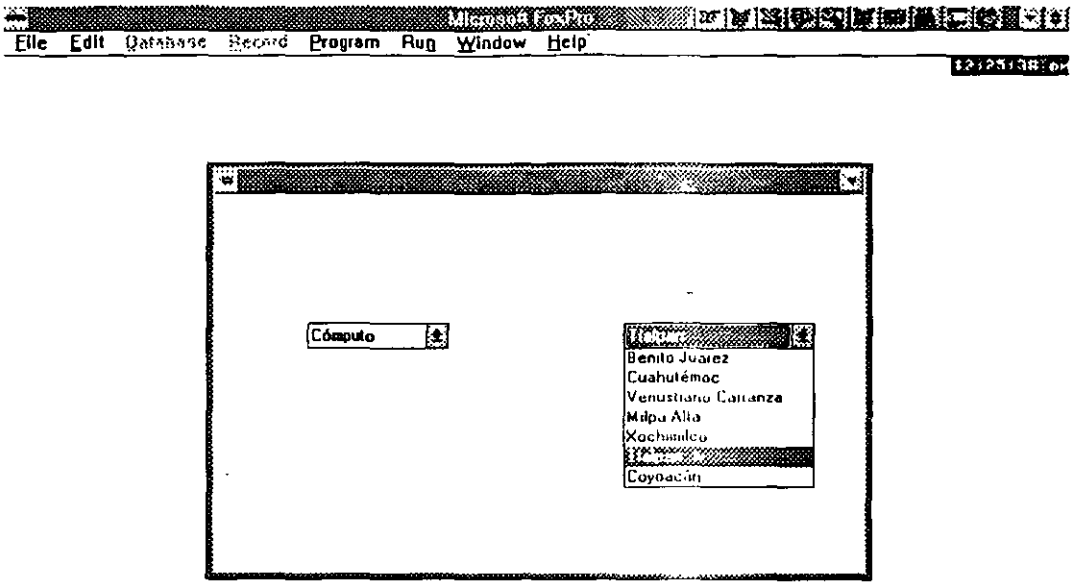


6.5 Controles popup (Popup)

6.5.1 Definición

Este tipo de objetos se refiere al tipo de listas que encontramos con frecuencia en Windows, en donde se nos presenta un marco con opciones en una lista desplegable.
Esta lista se nos presentará en primera instancia como un recuadro, mostrándonos el valor que se definió como default o la primera opción a seleccionar dentro de la lista, para accesar a los demas valores bastará con oprimir la flecha que aparece a la derecha del recuadro y nos mostrará el resto de la lista. Para el caso de que las opciones sobrepasen el tamaño del recuadro existirá una flecha de desplazamiento de la información.

Esto se verá de la siguiente forma:



El sistema nos definirá la opción sobre la que se encuentra el cursor mediante la barra iluminada y para seleccionar bastará oprimir <Enter> o el botón del mouse.

6.5.2 Sintaxis

Al igual que los otros objetos el formato del popup se especificará dentro de la FUNTION o en el PICTURE. Para indicarnos de que objeto se trata lo hará mediante un acento circunflejo (^).

Ejemplo:

```
@ 7.385,62.400 GET delega ;
PICTURE "@^ Benito Juárez;Cuahutémoc;Venustiano Carranza;Milpa
Alta;Xochimilco;Tlalpan;Coyoacán" ;
SIZE 1.538,23.833 ;
DEFAULT "Tlalpan" ;
FONT "MS Sans Serif", 8 ;
```

```
STYLE "B"

@ 7.385,14.400 GET inve ;
PICTURE "@^ Cómputo;Administración;Contabilidad;Finanzas" ;
SIZE 1.538,18.000 ;
DEFAULT "Cómputo" ;
FONT "MS Sans Serif", 8 ;
STYLE "B"
```

Para el caso del popup es posible crearlo a partir de un arreglo, esto nos será de gran utilidad ya que este objeto se convierte en dinámico, debido a que si por ejemplo, lo llenamos con los nombres de los departamentos de la empresa y el dia de mañana se crean dos y desaparece uno, bastará con actualizar la base de datos y no siendo necesario modificar el programa.

6.5.3 Popups a partir de arreglos

Cuando se utilizan arreglos para trabajar con popups se deberá de haber llenando previamente el arreglo ya sea elemento por elemento o mediante un proceso que extraiga la información de una base de datos.
La sintaxis en el momento de la construcción

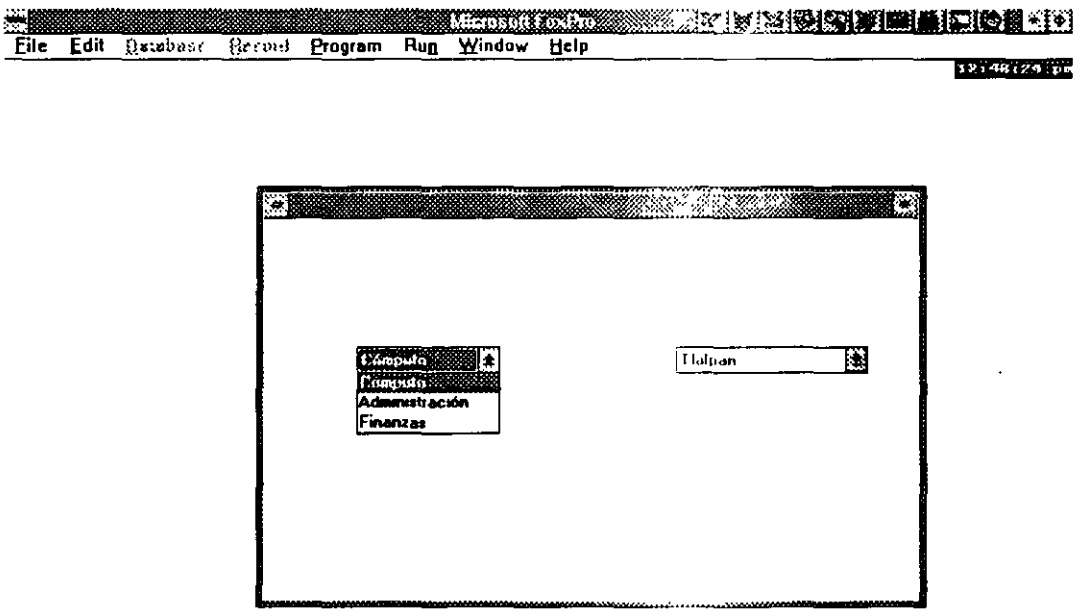
@ren,col GET <variable> PICTURE "@^" FROM <nombre del arreglo>

Es importante mencionar que si el arreglo fue definido de un tamaño y los elementos de la base son menos o el arreglo posee elementos vacíos se desplegarán los espacios en el POPUP.
Para el caso de querer solo desplegar un rango de elementos dentro del popup se utilizará el calificador RANGE, en donde le definiremos a partir de que elemento hasta cuál se mostrará.

Ejemplo:

```
DECLARE DEPTOS(5)
DEPTOS(1)="Cómputo"
DEPTOS(2)="Administración"
DEPTOS(3)="Finanzas"
DEPTOS(4)="Investigación"
DEPTOS(5)="Contabilidad"
@ 7.385,14.400 GET inve ;
     PICTURE "@^" from deptos;
     SIZE 1.538,18.000 ;
     DEFAULT "Cómputo" ;
     range 1,3;
     FONT "MS Sans Serif", 8 ;
     STYLE "B"
```

Se muestra el efecto de las instrucciones arriba mostradas en la siguiente figura:



6.6 Cajas de selección (Check Boxes)

6.6.1 Definición

Este objeto consta de cajas de selección donde las opciones seleccionadas serán aquellas que contengan dentro de la caja una X. Este objeto lo construiremos como elementos por separado esto es, cada caja de selección es un objeto por separado, a diferencia de los radio botones y los push buttons que todos los elementos conforman al objeto.
A diferencia de los radio botones en este caso podemos tener mas de una selección simultánea

6.6.2 Sintaxis

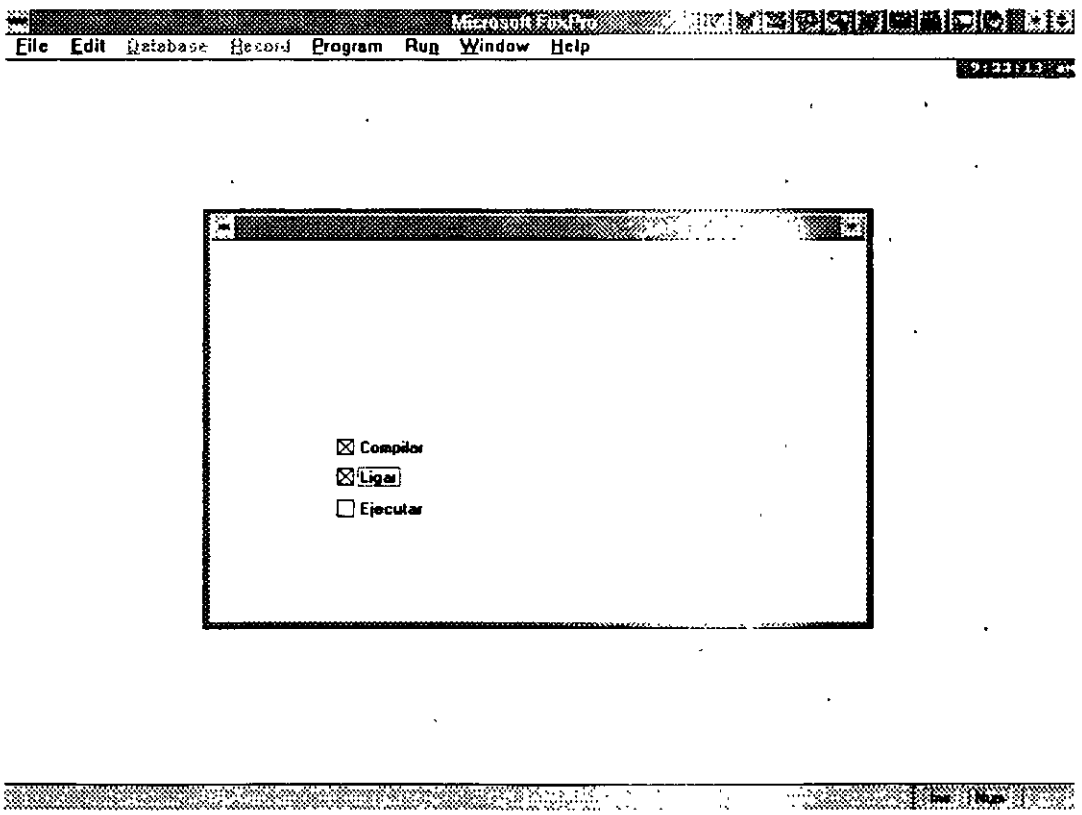
Denotaremos dentro de la sintaxis cuando se trabaja con cajas de selección porque contiene dentro de FUNCTION o PICTURE un * seguido de una C mayúscula.
@ren,col GET <variable> FUNCTION "*C <valor>" | PICTURE "@*C <valor>";
DEFAULT #;
SIZE <# altura>,<# ancho>,<# distancia>;
MESSAGE <mensaje>;
VALID <expresión que se ejecutará cuando la opción valida una condición>;
WHEN <expresión que se ejecutará bajo alguna condición>;
COLOR SCHEME <lista de colores>

Ejemplo:

```
@ 11.231,19.200 GET compila ;
     PICTURE "@*C Compilar" ;
     SIZE 1.308,12.500 ;
     DEFAULT 0 ;
     FONT "MS Sans Serif", 8 ;
     STYLE "BT"
@ 13.000,19.200 GET liga ;
```

```
        PICTURE "@*C Ligar" ;
        SIZE 1.308,9.167 ;
        DEFAULT 0 ;
        FONT "MS Sans Serif", 8 ;
        STYLE "BT"
@ 14.846,19.200 GET eje ;
        PICTURE "@*C Ejecutar" ;
        SIZE 1.308,12.333 ;
        DEFAULT 0 ;
        FONT "MS Sans Serif", 8 ;
        STYLE "BT"
```

Como vemos en la sintaxis cada caja representa una parte del programa no se encuentran dentro de la misma instrucción todas las opciones, lo que nos permite la selección múltiple.
En este caso el sistema pregunta si se desea Compilar, Ligar y Ejecutar un programa, como vemos en la figura sólo se compilará y ligará.



6.7 Listas desplegables (Scrolling List)

6.7.1 Definición

Las listas desplegables son objetos semejantes a los POPUPS con la característica de que aquí forsozamente se trabajará con arreglos o popups previamente definidos para llenar la lista. En este caso aparecerá dentro de un marco como área de despliegue definida por el usuario las opciones que se tienen, a diferencia del POPUP donde aparecía sólo la opción por default y había que abrir el objeto para ver las demás opciones.

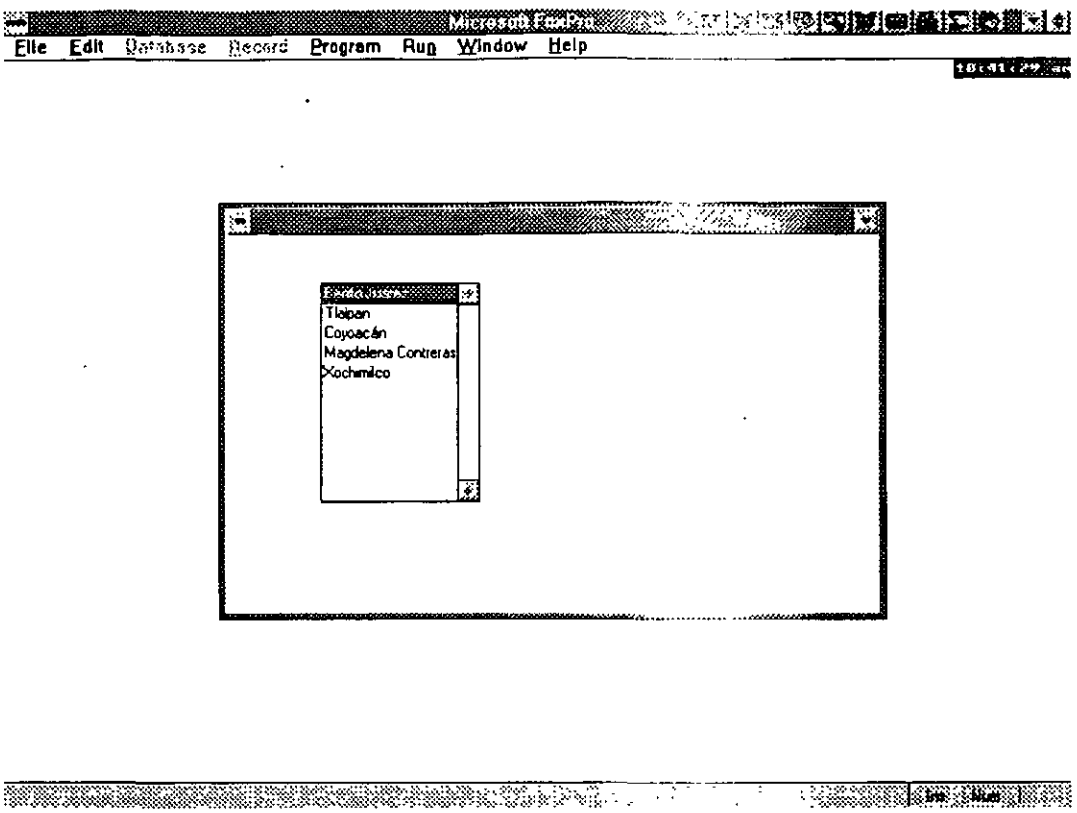
6.7.2 Sintaxis

Encontraremos dentro de la sintaxis un & para determinarle que se trata de una lista desplegable y seguiremos los mismos lineamientos del objeto POPUP cuando se trabaja con arreglos.

@ren,col GET <variable> PICTURE "@&" FROM <nombre del arreglo>

Ejemplo:

```
declare delega(5)
delega(1)="Benito Juárez"
delega(2)="Tlalpan"
delega(3)="Coyoacán"
delega(4)="Magdelena Contreras"
delega(5)="Xochimilco"
@ 2.846,14.600 GET var_del ;
      PICTURE "@&T" ;
      FROM delega ;
      SIZE 12.692,23.800 ;
      DEFAULT 1 ;
      FONT "MS Sans Serif", 8
```



En este caso vemos que el área que se le definió a la lista es mayor que el número de opciones existentes, por lo que muestra el resto de la lista en blanco. Cuando el número de opciones es mayor al área de despliegue el usuario podrá desplazarse mediante las flechas encontradas del lado derecho de la lista.

6.8 Contadores (Spinners)

6.8.1 Definición

Los Spinners son objetos contador, son aquellos donde su única función consiste en contar dentro de ciertos límites con un determinado incremento. Por ejemplo si se deseara que la edad de los empleados apareciera en una forma de captura y se desea evitar que el usuario capture le definiremos un spinner para que mediante éste seleccione ayudado de las flechas o el mouse la edad deseada.

6.8.2 Sintaxis

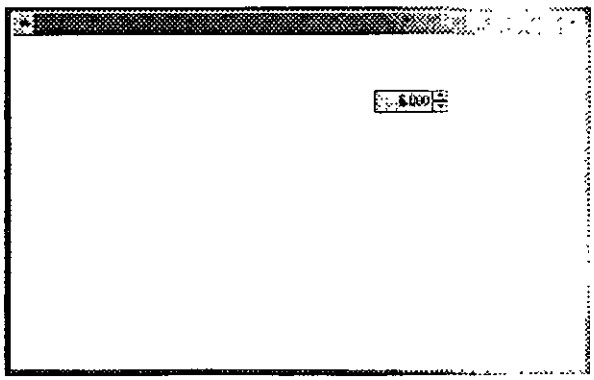
En este caso la sintaxis involucrará dentro del PICTURE o FUNCTION la letra K mayúscula, lo cual indica que se trata de un SPINNER.

```
@ ren,col GET <variable> ;
      SPINNER <lim. inferior>,<incremento>,<lim. superior>;
      PICTURE "@K" ;
      SIZE <alto>,<ancho> ;
      DEFAULT #;
      FONT <tipo de font>,<tamaño del font>
```

Aquí le definiremos a partir de que número iniciará, cuál es su límite y de cuanto en cuanto serán los incrementos.

Ejemplo:

```
@ 3.923,63.000 GET cuanto ;
      SPINNER 1.000, 1, 100 ;
      PICTURE "@K" ;
      SIZE 1.077,9.00 ;
      DEFAULT 1.000 ;
      FONT "MS Sans Serif", 8
```



Ejercicio

Para analizar nuestro aprendizaje en este capítulo se propone el siguiente ejercicio:

Crear las siguientes pantallas de captura con las siguientes características:

Altas de empleados donde:

1. En la dirección se tendrá un menú popup con todas las delegaciones existentes en el D.F.

2. En el Estado se tuviera un menú popup con los estados dejando como default el D.F.

3. La edad del empleado con un spinner iniciando en 18 y finalizando en 80.

4. Definir el tipo de empleado Sindicalizado, De confianza, Directivo por medio de radio botones

5. Definir las características del empleado Jefatura, Mando medio, Con subordinados, Sin subordinados, Empleado, con secretaria, sin secretaria, por medio de cajas de chequeo.

6. Por medio de botones con imágenes definir si es correcta o incorrecta la información y si desea grabarla o no.

7. Creación y manipulación de menús y popups

7.1 Definición

Se define como menú a la serie de instrucciones que se utilizan para crear un sistema de selecciones u opciones concatenadas o con relación entre ellas que nos permitirán elegir la ruta que se quiera seguir en el trabajo con el sistema. Para recorrerlo lo haremos mediante las flechas de selección o el mouse y le indicaremos al sistema cuál es la opción seleccionada oprimiendo el <enter> del teclado o del mouse.

Existen cuatro pasos dentro de la programación en cuanto al manejo de menús. Estos son los siguientes:

- Definición del menú y su estructura:

  DEFINE MENU <NOMBRE>

- Definición de las opciones o pads

  DEFINE PAD <NOMBRE> OF <NOMBRE_MENU> PROMPT <NOMBRE_OPCIÓN>

- Definición del Popup a activar dependiendo de la selección

  ON PAD <NOMBRE> OF <NOMBRE_MENU> ACTIVATE POPUP <NOMBRE_POP>

- Definición de cada popup o menú colgante

  DEFINE POPUP <NOMBRE_POP> FROM <REN,COL>

- Definición de cada barra de cada uno de los menús colgantes.
  DEFINE BAR # OF <NOMBRE_POP> PROMPT <NOMBRE_OPCIÓN>
- Definición de la acción a seguir dependiendo de la selección hecha.

ON SELECTION POPUP <NOMBRE_POP> DO <NOMBRE_PROC> WITH POPUP(),PROMPT()

- Activar el menú

ACTIVATE MENU <NOMBRE>

Ejemplo:

Definiremos el menú en la línea 0, horizontal.

DEFINE MENU MENU_PRI BAR AT LINE 0
Ahora se definirán las opciones que lo componen

DEFINE PAD primero OF menu_pri PROMPT [\<ALTAS]
DEFINE PAD segundo OF menu_pri PROMPT [\<BAJAS]
DEFINE PAD tercero OF menu_pri PROMPT [\<MODIFICACIONES]
DEFINE PAD cuarto OF menu_pri PROMPT [\<CONSULTAS]
DEFINE PAD quinto OF menu_pri PROMPT [\<SALIR]

Una vez creadas las opciones se determinará la acción a seguir dependiendo de la selección hecha por el usuario.

ON PAD primero OF menu_pri ACTIVATE POPUP menu1
ON PAD segundo OF menu_pri ACTIVATE POPUP menu2
ON PAD tercero OF menu_pri ACTIVATE POPUP menu3
ON PAD cuarto OF menu_pri ACTIVATE POPUP menu4

Ya definido lo que sucederá al seleccionar una u otra opción del menú se indicará la siguiente acción, que en este caso representa el despliegue de un menú colgante.

A continuación se definen los menús colgantes de cada opción.

DEFINE POPUP menu1 FROM 1,1
DEFINE POPUP menu2 FROM 1,10

```
DEFINE POPUP menu3 FROM 1,20
DEFINE POPUP menu4 FROM 1,35
```

Se determinará donde se desea que inicie el despliegue del menú, seguido de el contenido de éste barra por barra.

```
DEFINE BAR 1 OF menu1 PROMPT [\<Empleado]
DEFINE BAR 2 OF menu1 PROMPT [\<Departamentos]
DEFINE BAR 3 OF menu1 PROMPT [\<Categorías]
DEFINE BAR 4 OF menu1 PROMPT [\<Salir]
```

Siendo como último escalón de esta escalera de menús el llamar algún procedimento se ejecute dependiendo de la opción hecha.

Se enviarán como datos a la subrutina el nombre del popup y nombre de la opción elegida.

```
ON SELECTION POPUP menu1 DO altas WITH POPUP(),PROMPT()

DEFINE BAR 1 OF menu2 PROMPT [\<Empleado]
DEFINE BAR 2 OF menu2 PROMPT [\<Departamentos]
DEFINE BAR 3 OF menu2 PROMPT [\<Categorías]
DEFINE BAR 4 OF menu2 PROMPT [\<Salir]
ON SELECTION POPUP menu2 DO bajas WITH POPUP(),PROMPT()

DEFINE BAR 1 OF menu3 PROMPT [\<Empleado]
DEFINE BAR 2 OF menu3 PROMPT [\<Departamentos]
DEFINE BAR 3 OF menu3 PROMPT [\<Categorías]
DEFINE BAR 4 OF menu3 PROMPT [\<Salir]
ON SELECTION POPUP menu3 DO modif WITH POPUP(),PROMPT()

DEFINE BAR 1 OF menu4 PROMPT [\<Empleado]
DEFINE BAR 2 OF menu4 PROMPT [\<Departamentos]
DEFINE BAR 3 OF menu4 PROMPT [\<Categorías]
DEFINE BAR 4 OF menu4 PROMPT [\<Salir]
ON SELECTION POPUP menu4 DO consul WITH POPUP(),PROMPT()
```

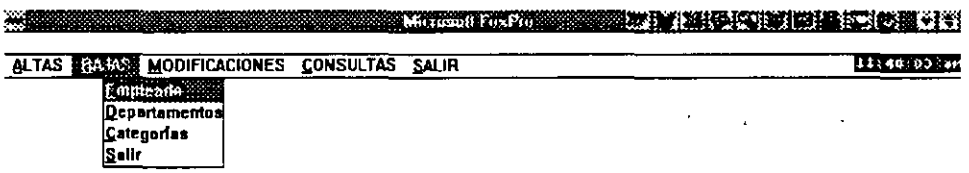
Una vez que está todo definido se activa el menú principal.

```
ACTIVATE MENU menu_pri
Return
```

Veremos la subrutina

```
PROCEDURE ALTAS
PARAMETERS nomb_men,nom_op
Do case
   case nom_op="Empleados"
   .
   case nom_op="Departamentos"
   .
Endcase
```

La definición del sistema de menús creado anteriormente nos crea la siguiente pantalla:



Ejercicio

Para verificar que efectivamente el capítulo cumplió con su objetivo de que el usuario sea capaz de crear un sistema de menús realizaremos el siguiente ejercicio.

1. Crear el menú principal del sistema que contenga las siguientes características:

2. Las opciones deberán de ser Empleados, Tabulador, Nómina, Departamentos, Procesos, Utilerías y Salir.

3. Cada opción poseerá un menú colgante con las opciones válidas dentro de cada rubro, incluyendo impresiones.

4. Dentro de impresiones se deberá de tener para el caso de Nómina varias opciones en donde se tenga: Impresiones de la nómina por diferente formas de ordenamiento (por departamento, por no. de empleado, etc.)

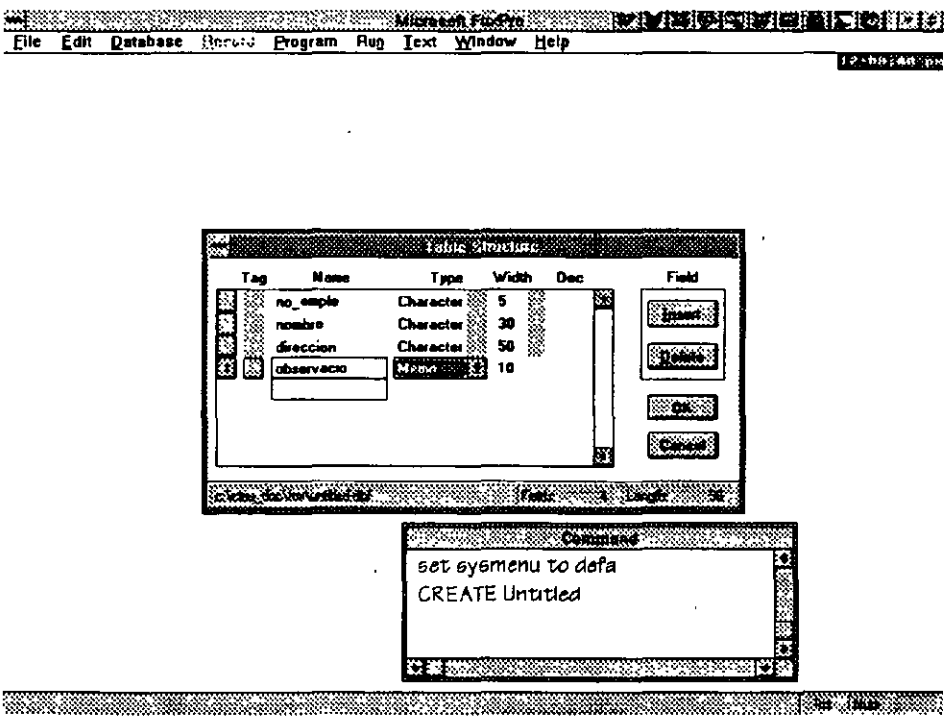
5. Este sistema de menú deberá de ser cíclico y poder regresar de menú en menú

8. Variables de memoria

8.1 Definición

Definiremos campo memo a un espacio de memoria donde se almacenará información, las diferencias entre este y una variable de tipo caracter será en primer lugar que la primera es de longitud fija y el segundo no, el campo memo varía según el porcentaje de registros ocupados y la longitud del campo memo en cada registro que puede ser mayo a 254 Kb, la única restricción es el espacio libre en disco que posea. Cuando es creado un campo memo dentro de una base de datos se generará automáticamente un archivo llamado igual que la base de datos pero con extensión FPT, donde guardará la información de todos los campos memo, si este archivo es borrado no será posible accesar a la base. Por lo que se recomienda tener especial cuidado en este aspecto.

Definiremos dentro de nuestra base de datos un campo de tipo memo como se muestra en la figura:

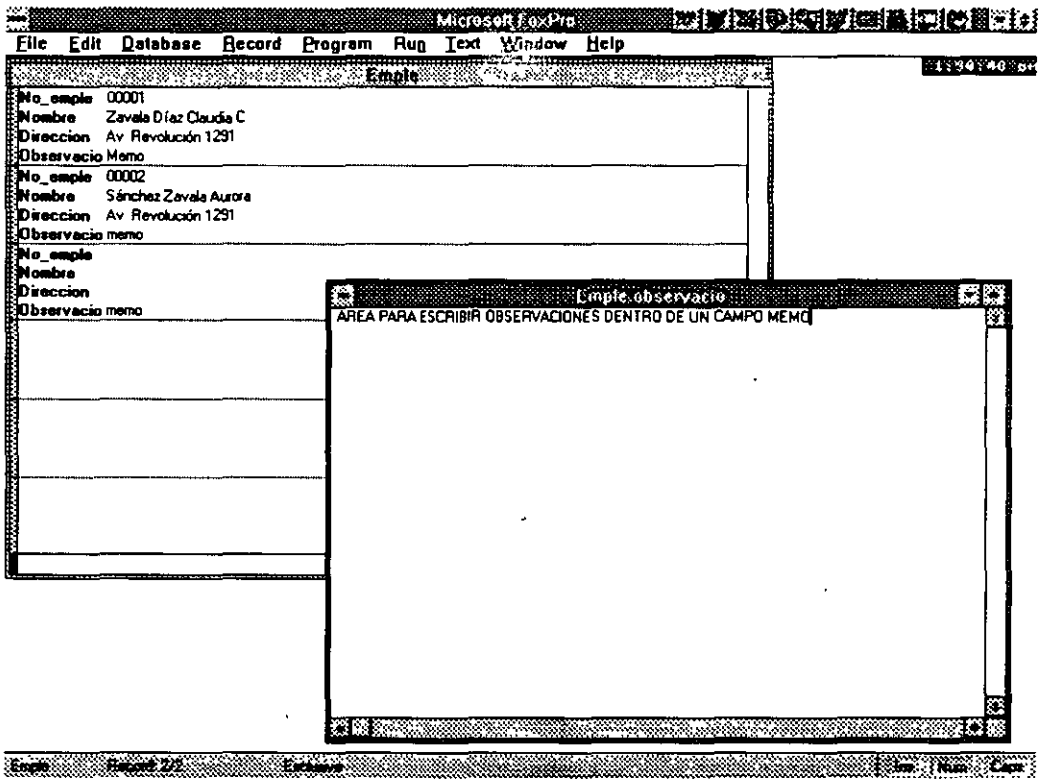


Como vemos le asignará por default un tamaño de 10.

8.2 Operaciones con campos memo

8.2.1 Editar

Para editar un campo memo desde un browse, veremos que en el despliegue de la base de datos el campo memo sólo se ve la palabra "memo", para abrirlo bastará darle dos clicks con el mouse o <CTRL> <Avpag> y abrirá el espacio para poder trabajar con él. Si se desea salir salvando se oprimirá <Ctrl><W> y salir sin salvar <Esc>.



Se podrá notar que una vez que se teclea información en un campo memo cambiará la primera m de la palabra a M mayúscula.

8.2.2 Modificar

Aún cuando desde el browse es posible modificar el contenido de un campo memo lo veremos desde el punto de vista de la programación.

```
USE EMPLEADO INDEX EMPLEADO
SEEK "00045"
MODIFY MEMO OBSERVA
```

8.2.3 Leer

Para poder extraer el contenido de un campo memo con una longitud determinada se hará lo siguiente:

Dividir el campo memo en líneas de long. deseada con la instrucción:

```
SET MEMOWIDTH TO #
```

donde #: No. de caracteres por línea

Después conoceremos el no. de líneas que existen dentro del memo de longitud determinada en el registro seleccionado, mediante la instrucción:

```
VAR_NUM=MEMLINES(<CAMPO MEMO>)
```

y su contenido

```
LINEA=MLINE(<CAMPO>,<#LINEA>)
```

Para ver esto dentro de un programa tenemos:

Ejemplo:

```
USE DATOS INDEX DATOS
SET MEMOWIDTH TO 20
SEEK "00051"
NO_LINES=MEMLLINES(OBSERVA)
FOR I=1 TO N_LINES
    LINEA=MLINE(OBSERVA,I)
    @ I,40 SAY LINEA
```

```
NEXT
```

8.2.4 Copia a un archivo

Se selecciona el registro a enviar a un archivo de tipo texto y se le dará la siguiente instrucción:

```
COPY MEMO <CAMPO> TO <ARCHIVO>
```

Ejemplo:

```
USE DATOS INDEX DATOS
SEEK "00089"
COPY MEMO OBSERVA TO ARCH.TXT
```

8.2.5 Transferir un archivo a un campo memo

Se deberá de buscar el registro al cuál se le anexará la información contenida en un archivo de texto, esto con la instrucción:

```
APPEND MEMO <CAMPO MEMO> FROM <ARCHIVO>
```

quedando:

Ejemplo:

```
USE DATOS INDEX DATOS
SEEK "00090"
APPEND MEMO OBSERVA FROM ARCH.TXT
```

Lo que hará que dentro del campo memo de la base de datos se almacenará el contenido de un archivo de texto.

8.2.6 Ventana de despliegue del memo

Para restingir el despliegue del campo memo dentro de una ventana se utilizará el siguiente comando:

SET WINDOW OF MEMO TO <VENTANA>

En este case el tipo de ventana se deberá de haber predefinido previamente.

Ejemplo:

```
DEFINE WINDOW WIN_MEMO FROM 17,20 TO 23,75 DOUBLE
ACTIVATE WINDOW WIN_MEMO
SET WINDOW OF MEMO TO WIN_MEMO
```

Ejercicio

Mediante la programación realice las siguientes actividades:

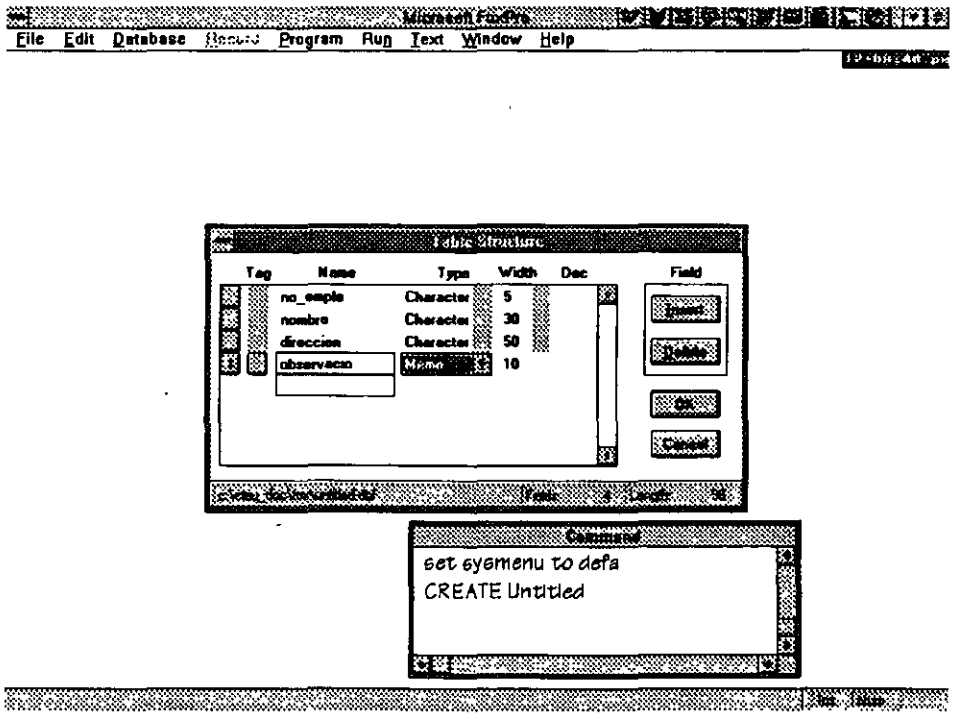
1. Dé de alta datos en el campo memo
2. El tamaño de las líneas del campo memo deberán de ser de 25 caracteres por línea
3. Mostrar en pantalla el campo memo dentro de una ventana delimitada en color diferente al resto de la pantalla
4. Grabe en un archivo de texto el contenido de por lo menos 10 campos memo.
5. Cree un archivo llamado memo.txt y guárdelo en el registro 10 de la base de datos, dentro del campo memo.

8. Variables de memoria

8.1 Definición

Definiremos campo memo a un espacio de memoria donde se almacenará información, las diferencias entre este y una variable de tipo caracter será en primer lugar que la primera es de longitud fija y el segundo no, el campo memo varía según el porcentaje de registros ocupados y la longitud del campo memo en cada registro que puede ser mayo a 254 Kb, la única restricción es el espacio libre en disco que posea. Cuando es creado un campo memo dentro de una base de datos se generará automáticamente un archivo llamado igual que la base de datos pero con extensión FPT, donde guardará la información de todos los campos memo, si este archivo es borrado no será posible accesar a la base. Por lo que se recomienda tener especial cuidado en este aspecto.

Definiremos dentro de nuestra base de datos un campo de tipo memo como se muestra en la figura:

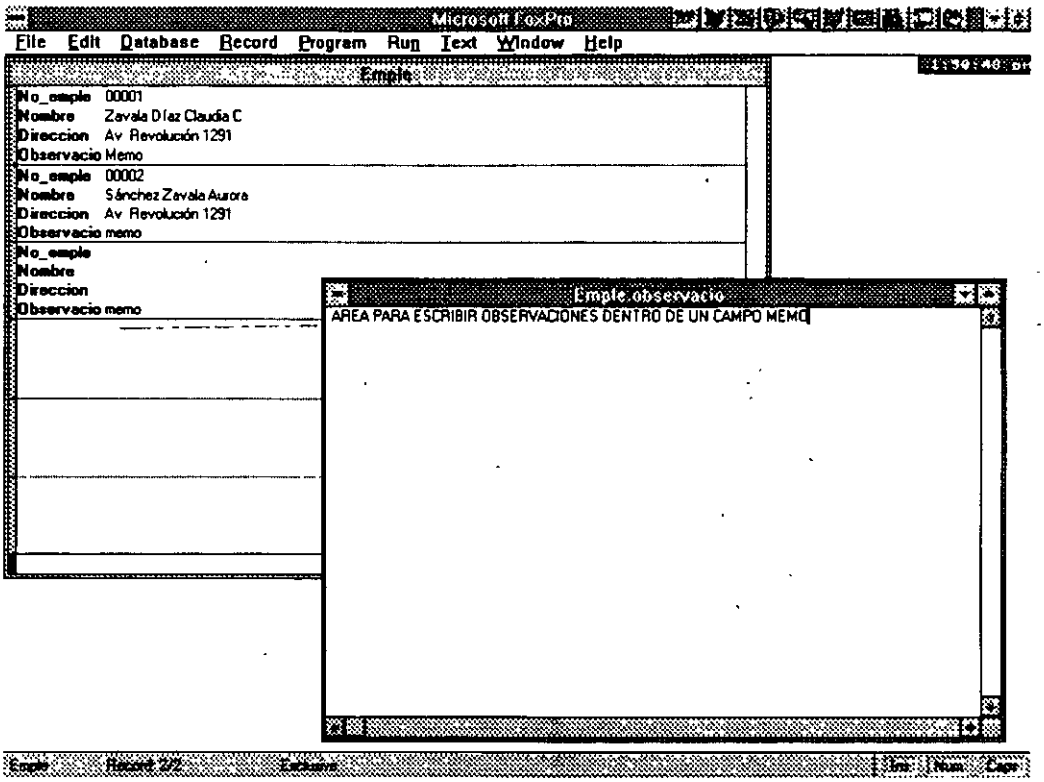


Como vemos le asignará por default un tamaño de 10.

8.2 Operaciones con campos memo

8.2.1 Editar

Para editar un campo memo desde un browse, veremos que en el despliegue de la base de datos el campo memo sólo se ve la palabra "memo", para abrirlo bastará darle dos clicks con el mouse o <CTRL> <Avpag> y abrirá el espacio para poder trabajar con él. Si se desea salir salvando se oprimirá <Ctrl><W> y salir sin salvar <Esc>.



Se podrá notar que una vez que se teclea información en un campo memo cambiará la primera m de la palabra a M mayúscula.

8.2.2 Modificar

Aún cuando desde el browse es posible modificar el contenido de un campo memo lo veremos desde el punto de vista de la programación.

```
USE EMPLEADO INDEX EMPLEADO
SEEK "00045"
MODIFY MEMO OBSERVA
```

8.2.3 Leer

Para poder extraer el contenido de un campo memo con una longitud determinada se hará lo siguiente:

Dividir el campo memo en líneas de long. deseada con la instrucción:

```
SET MEMOWIDTH TO #
```

donde #: No. de caracteres por línea

Después conoceremos el no. de líneas que existen dentro del memo de longitud determinada en el registro seleccionado, mediante la instrucción:

```
VAR_NUM=MEMLINES(<CAMPO MEMO>)
```

y su contenido

```
LINEA=MLINE(<CAMPO>,<#LINEA>)
```

Para ver esto dentro de un programa tenemos:

Ejemplo:

```
USE DATOS INDEX DATOS
SET MEMOWIDTH TO 20
SEEK "00051"
NO_LINES=MEMLLINES(OBSERVA)
FOR I=1 TO N_LINES
    LINEA=MLINE(OBSERVA,I)
    @ I,40 SAY LINEA
```

```
NEXT
```

8.2.4 Copia a un archivo

Se selecciona el registro a enviar a un archivo de tipo texto y se le dará la siguiente instrucción:

```
COPY MEMO <CAMPO> TO <ARCHIVO>
```

Ejemplo:

```
USE DATOS INDEX DATOS
SEEK "00089"
COPY MEMO OBSERVA TO ARCH.TXT
```

8.2.5 Transferir un archivo a un campo memo

Se deberá de buscar el registro al cuál se le anexará la información contenida en un archivo de texto, esto con la instrucción:

```
APPEND MEMO <CAMPO MEMO> FROM <ARCHIVO>
```

quedando:

Ejemplo:

```
USE DATOS INDEX DATOS
SEEK "00090"
APPEND MEMO OBSERVA FROM ARCH.TXT
```

Lo que hará que dentro del campo memo de la base de datos se almacenará el contenido de un archivo de texto.

8.2.6 Ventana de despliegue del memo

Para restingir el despliegue del campo memo dentro de una ventana se utilizará el siguiente comando:

SET WINDOW OF MEMO TO <VENTANA>

En este case el tipo de ventana se deberá de haber predefinido previamente.

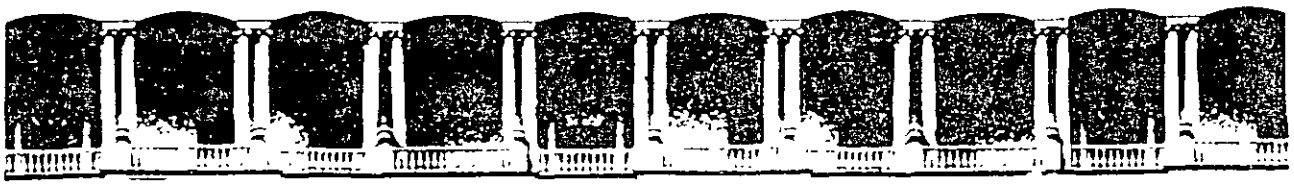
Ejemplo:

```
DEFINE WINDOW WIN_MEMO FROM 17,20 TO 23,75 DOUBLE
ACTIVATE WINDOW WIN_MEMO
SET WINDOW OF MEMO TO WIN_MEMO
```

Ejercicio

Mediante la programación realice las siguientes actividades:

1. Dé de alta datos en el campo memo
2. El tamaño de las líneas del campo memo deberán de ser de 25 caracteres por línea
3. Mostrar en pantalla el campo memo dentro de una ventana delimitada en color diferente al resto de la pantalla
4. Grabe en un archivo de texto el contenido de por lo menos 10 campos memo.
5. Cree un archivo llamado memo.txt y guárdelo en el registro 10 de la base de datos, dentro del campo memo.



FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

COMPLEMENTO DE MATERIAL DIDACTICO

FOX PRO BAJO AMBIENTE WINDOWS

JUNIO, 1996

Palacio de Minería Calle de Tacuba 5 Primer piso Deleg Cuauhtemoc 06000 México, D F APDO Postal
Teléfonos: 512-8955 512-5121 521-7335 521-1987 Fax 510-0573 512-5121 521-4020 AL 26

INDICE

INDICE

ÍNDICE

9. Imagen y sonido

9.1 Introducción

Para poder trabajar con imágenes en Fox pro será necesario retomar el concepto de campo genérico o general, los cuáles pueden contener cualquier tipo de dato: imagen, sonido, gráficas, tablas de una hoja de cálculo, esto será pegado a la base de datos mediante la capacidad DDE (Dynamic Data Exchange) y OLE (Object Linking and Embedding) de Windows.

Cualquier dato procedente de un paquete que trabaje en ambiente Windows que soporte datos DDE u OLE podrá ser incrustado en un campo genérico de una base de datos.

La inclusión de los datos se realiza por medio del portapapeles de Windows y la función cortar y pegar existente en todas las aplicaciones que trabajan en este ambiente.

Para poder entender con mayor precisión el capítulo, será necesario incluir los términos de vínculo e incrustación.

9.2 Vínculo e Incrustación

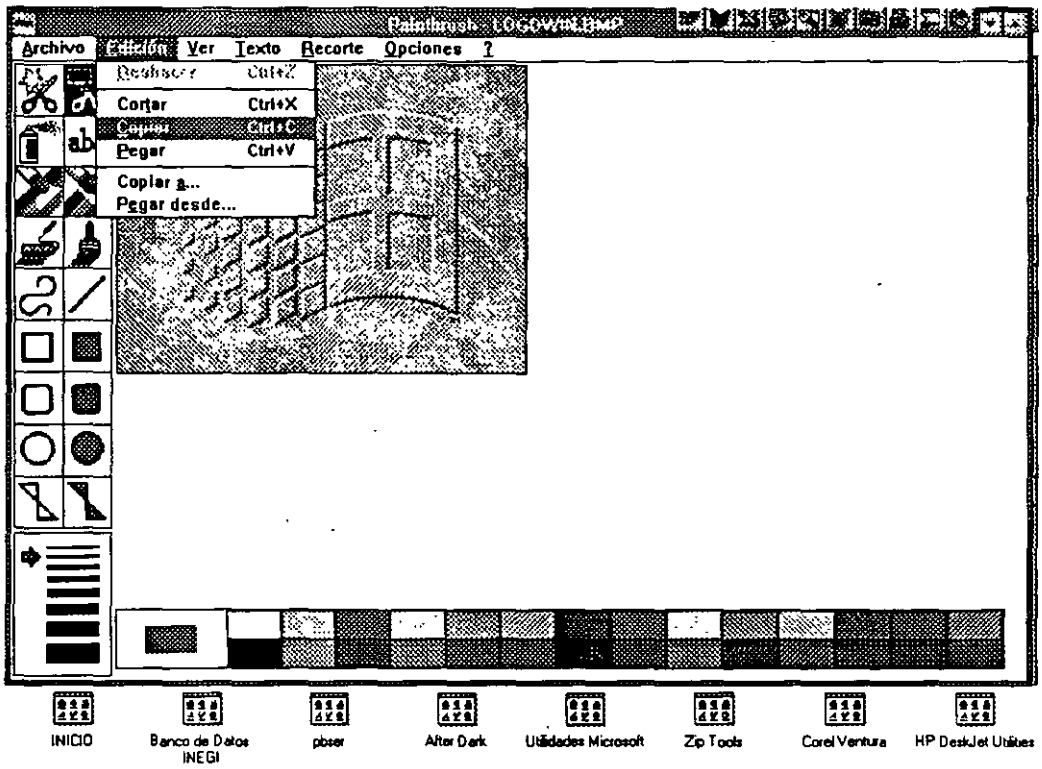
Llamaremos incrustación cuando el objeto que maneje Foxpro sea una copia del objeto creado en una determinada aplicación. Por ejemplo: Se crea una imagen en Paint Brush, si se incrusta una copia de ésta será incrustada a el campo genérico.

En el caso de vincular, la liga entre el objeto y su paquete de origen no se pierde, por lo que si se modifica el original se modificará el campo en la base de datos. Asimismo en el momento que quisiéramos modificar el objeto, llamaría directamente a la aplicación en la que fue elaborado.

9.3 Cómo añadir imagenes a una base de datos

1. Abrir una aplicación que contenga una imagen

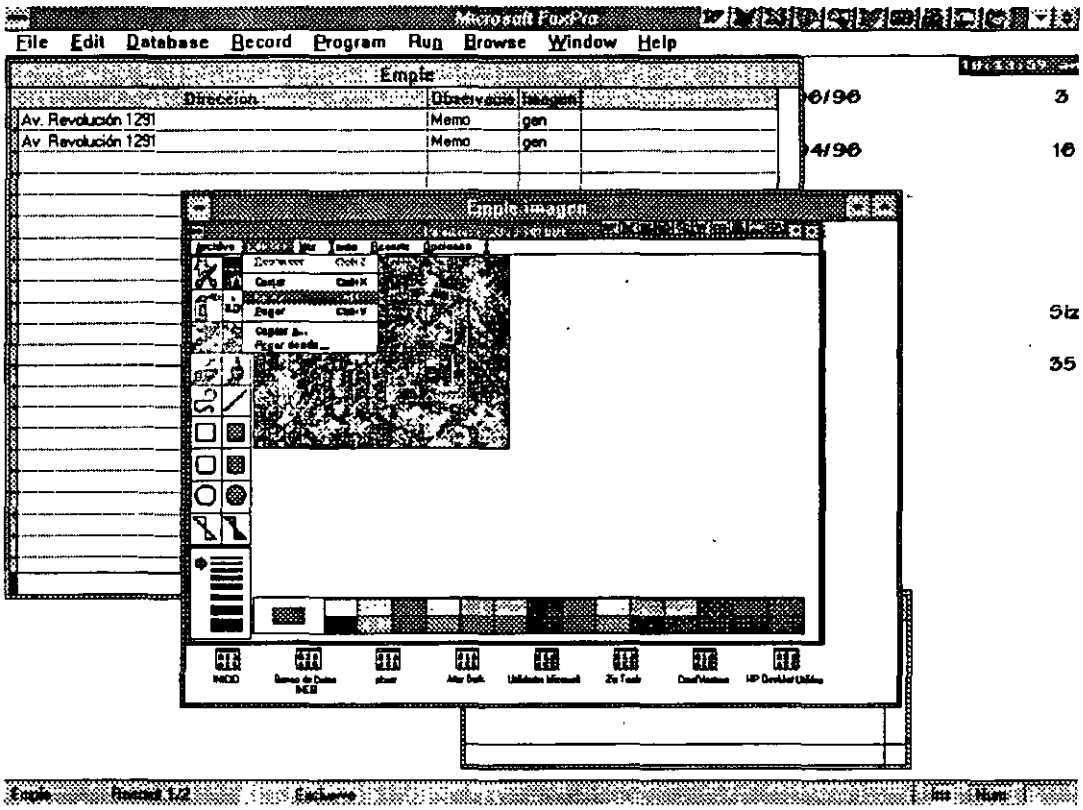
2. Seleccione la imagen completa o parte de ella y utilice Edición / Cortar o Edición/Copiar. En la siguiente pantalla se muestra un ejemplo tomando como aplicación a Paint Brush, se edito una imagen y se Copió al portapapeles



3. Se deberá de tener activo Foxpro de preferencia con la base de datos a la que se le incrustará el objeto abierta.

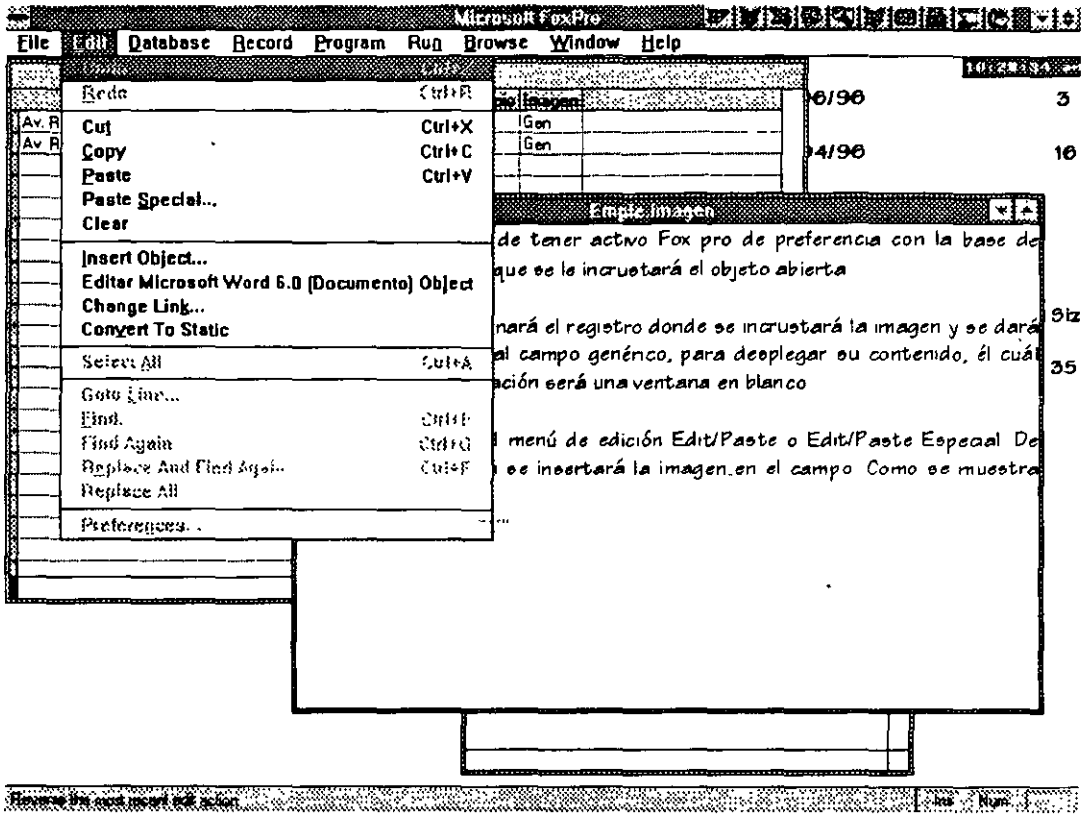
4. Se seleccionará el registro donde se incrustará la imagen y se dará dos clicks al campo genérico, para desplegar su contenido, él cuál en esta ocación será una ventana en blanco.

5. Se llama al menú de edición Edit/Paste o Edit/Paste Especial. De esta forma se insertará la imagen en el campo. Como se muestra en la figura.



De la misma forma trabajará con texto, hojas de cálculo, gráficas, etc., cualquier objeto OLE.

Para el caso de objetos ligados el mismo menú de edición nos permitirá cambiarnos a la aplicación correspondiente del objeto incrustado. En el siguiente ejemplo se vinculó un texto de Word, en este caso como fue vinculado y no incrustado no perderá la liga con el procesador hasta que el objeto se convierta en estático, esto mediante la opción Edit/Convert to static.



9.4 Visualización del contenido de un campo genérico

Para poder tener acceso a los datos de un campo genérico, se tratará a estos como cualquier dato al que se puede desplegar en una pantalla o pegar a un reporte, tan sólo con el comando SAY y el nombre del campo.

Ejercicio

1. Incluye en la base de datos un campo genérico que se llamará foto

2. Vincula en el segundo y tercer registro a la imagen

3. Incrusta en los demás datos las imagenes

4. En el último campo incluye una gráfica en Excel

5. En el primer registro vincula un texto de Word.

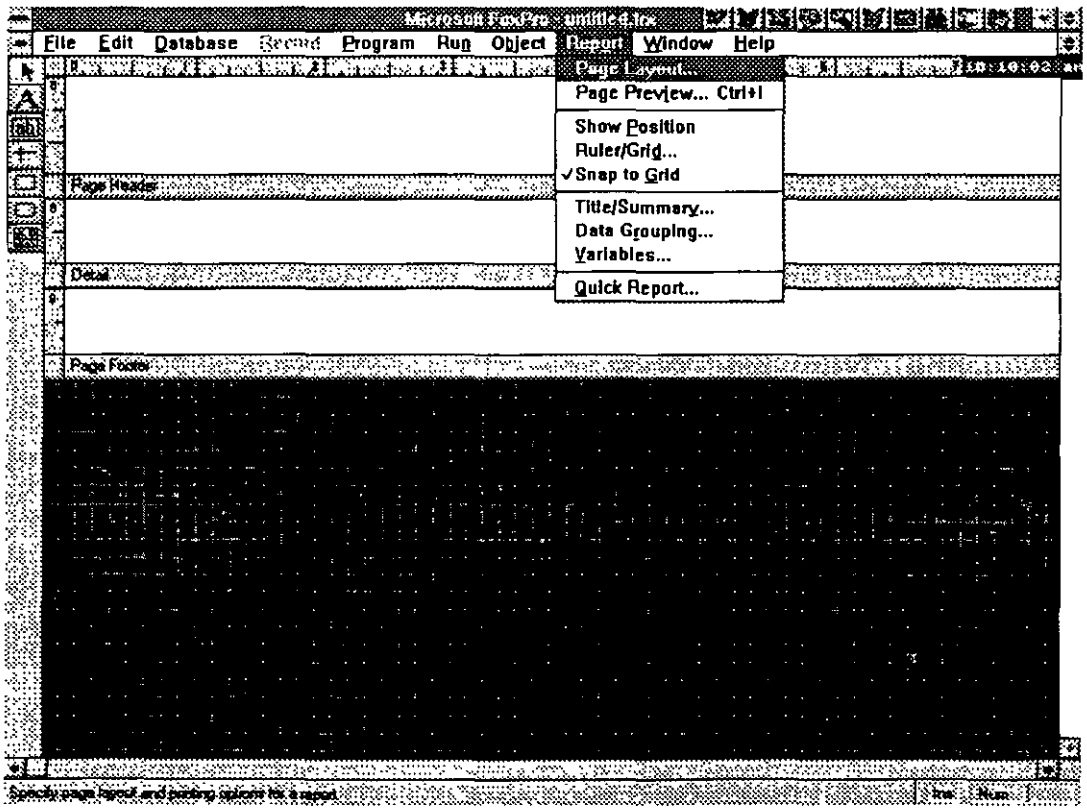
10. Reporteador

10.1 Definición

El generador de reportes es una herramienta que utilizaremos para crear reportes de nuestras bases de datos con gran diversidad, en la cuál será posible manipular la información agrupándola, obteniendo sumatorias, acumulados, etc.

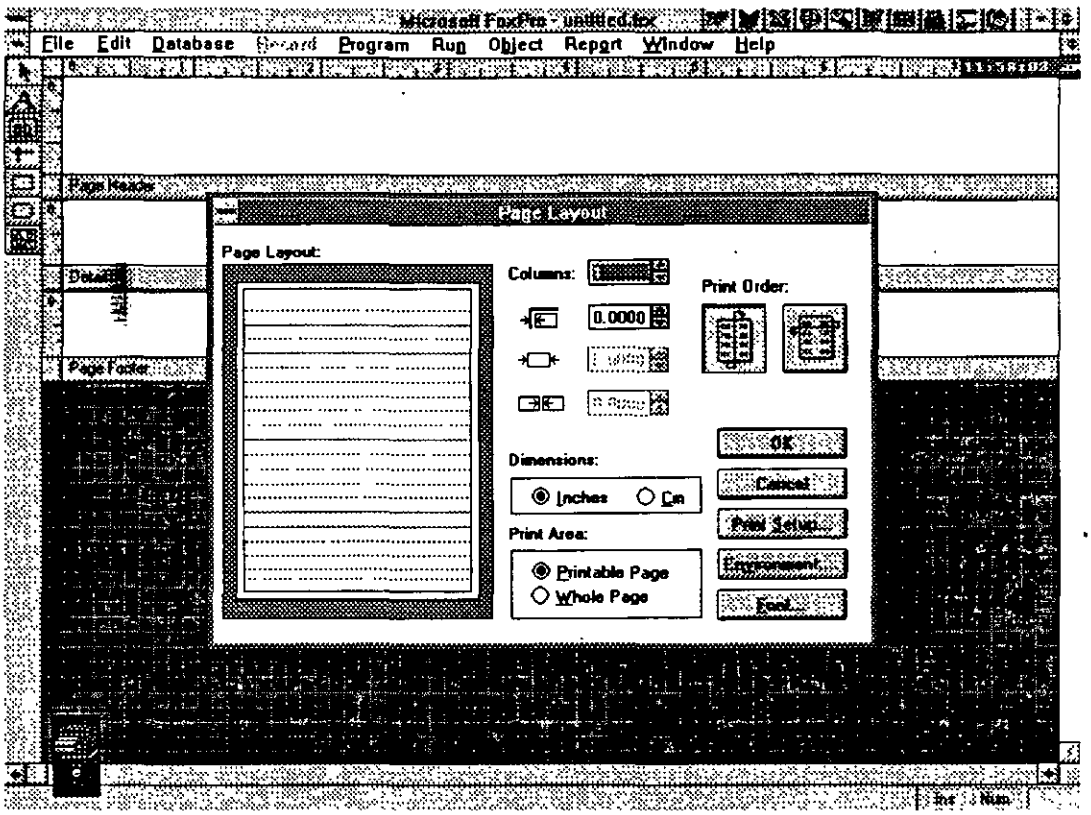
10.2 Generador de reportes

Llamaremos al generador por medio del menú File-New-Report y se añadirá una nueva opción en la barra del menú superior Report, presentándose la siguiente pantalla:



Dentro del menú desplegable que presenta la opción de Report tendremos las siguientes opciones que definiremos a continuación:

- **Page Layout:** Esta opción permite cambiar las especificaciones del diseño de la página. Al ser seleccionada nos presentará la siguiente pantalla de diálogo:



En donde:

Left margin: Está representado mostrando un recuadro con un margen izquierdo, se definirá aquí el espacio que tendrá la impresión con respecto a la orilla izquierda del papel.

Columns: Este valor define el número de columnas del reporte. Es posible elegir de 1 a 50 columnas

Column width: Aquí se le definirá el ancho que tendrán las columnas y se muestra como un recuadro y dos flechas una izquierda y la otra derecha limitándolo.

Space between columns: Espacio que existirá entre las columnas. Se lo definiremos en el área donde aparecen dos tipo de recuadros y unas flechas encontradas.

Dimensions: Define si los valores se tomarán en cm. o pulgadas.

Print area: En este apartado le definiremos al reporteador el área de impresión. Printable page, imprimirá dentro de los márgenes especificados y Whole page, tomará como área de impresión toda la hoja.

Print setup: Configura la impresora

Enviroment: Se utiliza para salvar, restaurar o limpiar el ambiente de trabajo actual.

Font: Tipo de letra que se utilizará en el reporte.

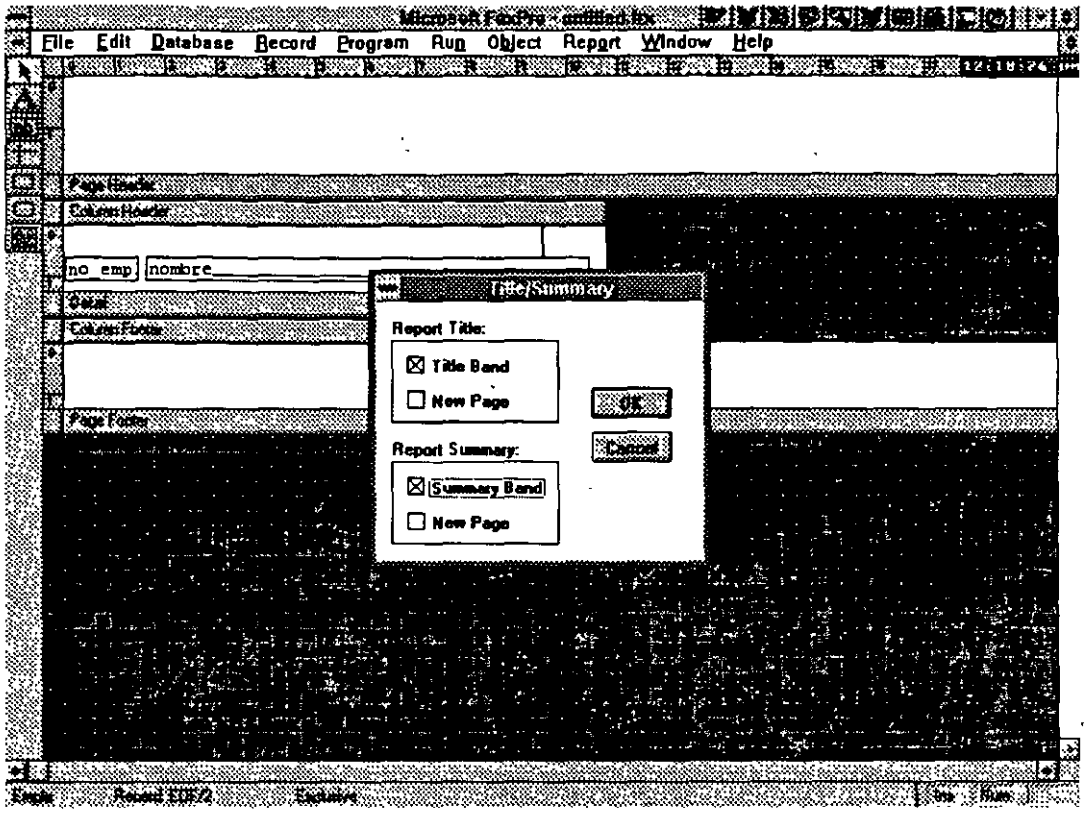
- **Page Preview:** Me permite obtener una presentación previa del aspecto del reporte. En el preview podremos ver hoja por hoja y realizar acercamientos (zoom in) y alejamientos (zoom out) de la hoja vista en pantalla.

- **Show Position:** Muestra o no en la barra de estado la posición que guarda el mouse o el cursor dentro de la pantalla

- **Ruler/Grid:** Gradúa la regla de la pantalla en cm, pulgadas o pixeles

- **Snap to Grid:** Permite que se trabaje o no con una línea invisible que alinea los campos puestos aparentemente en la misma posición dentro de la pantalla

- **Title/Summary:** Permite añadir bandas de títulos y sumarios en el reporte, al seleccionar esta opción aparecerá la siguiente pantalla:



Donde preguntará si se tendrán títulos y un resumen de lo hecho en el reporte en donde inicia y finaliza el reporte o en una hoja nueva.

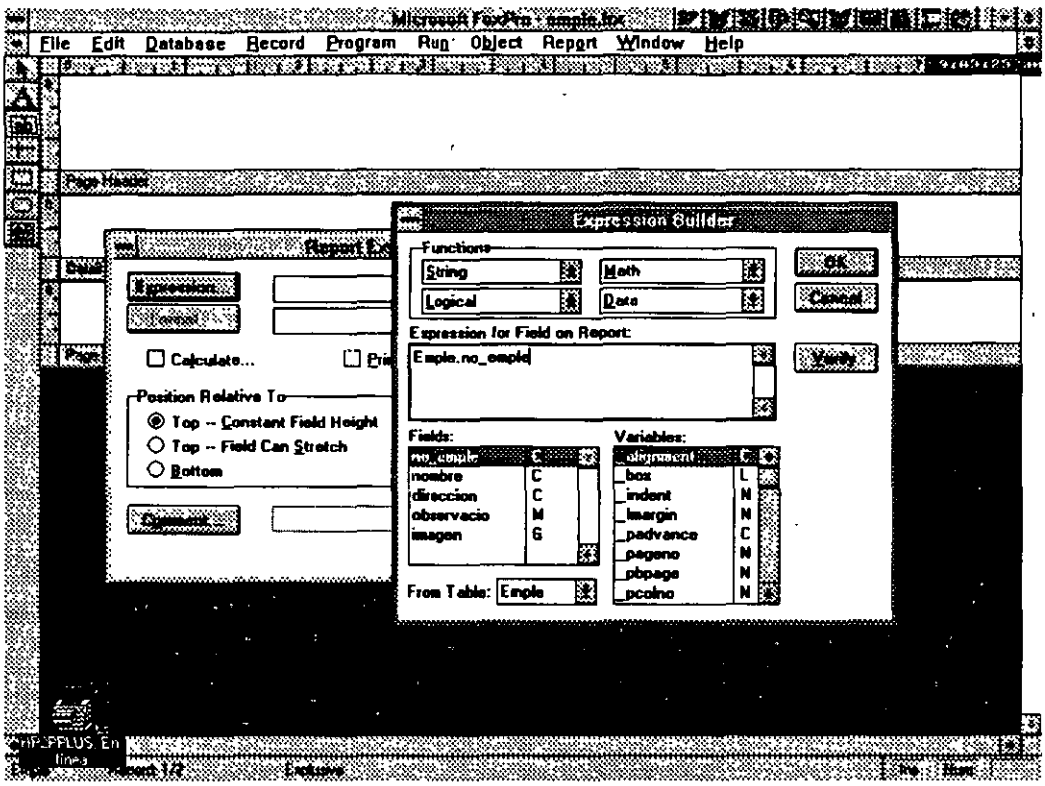
10.3 Barra de herramientas (Toolbox)

En la parte izquierda del reporte encontraremos botones con las herramientas que nos proporciona el constructor para ayudar a un rápido acceso a ciertos procesos.
Para insertar un objeto a un reporte ha de hacer click primero sobre la herramienta solicitada y después arrastrar hasta la ventana del reporte. Para incluir mas de un objeto del mismo tipo haga doble click en la barra de herramientas; ello le permitirá añadir múltiples objetos del mismo tipo.

10.3.1 Inserción de campos

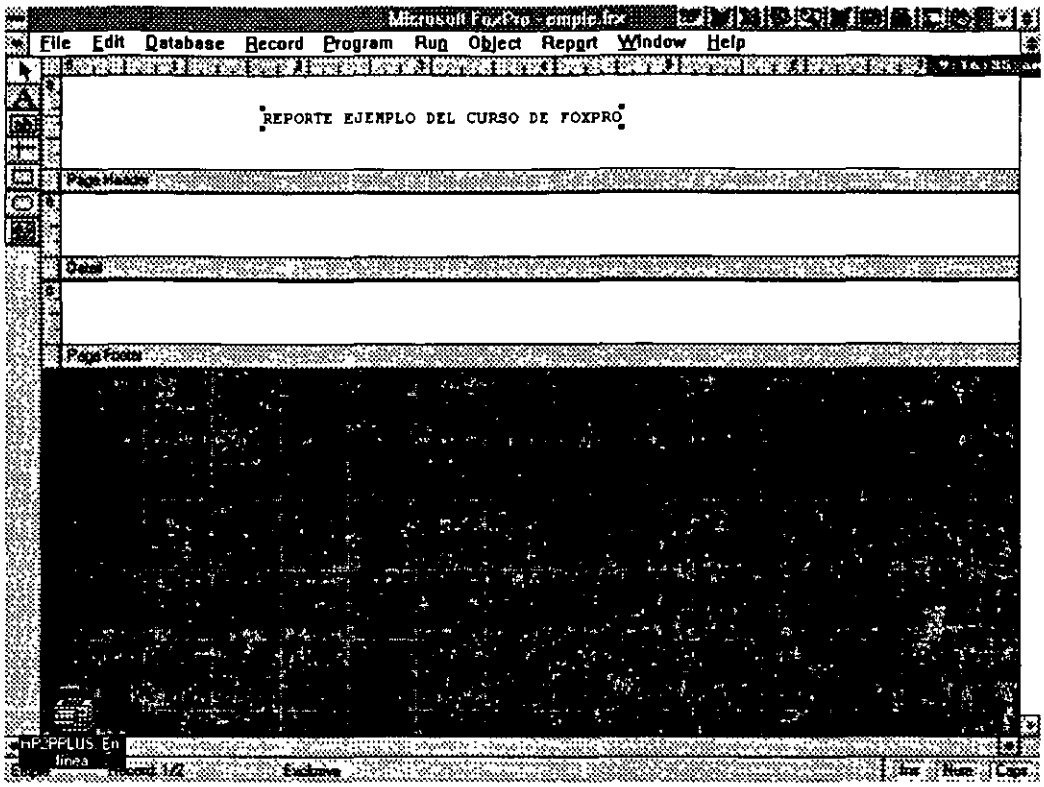
Para insertar un campo dentro del reporte lo haremos mediante el botón marcado con dos letras minúsculas **ab** y después en la ventana del reporte marcar en donde se desee colocar. Aparecerá

una caja de diálogo en donde por medio del botón Expression se podrá seleccionar el campo de todos los existentes dentro de la base de datos y finalmente OK.



10.3.2 Añadir texto

Para incluir texto dentro del reporte se deberá de oprimir el botón marcado con una **A** mayúscula, seleccionar a partir de donde se desea iniciar a escribir dentro del cuerpo del reporte y hacerlo, cuando se desea dejar de hacerlo se deberá de oprimir nuevamente este botón. El texto se podrá mover, para ello se deberá de seleccionar primeramente el texto por medio de el click del mouse, después de esto aparecerá un marco, por medio de las flechas de desplazamiento o arrastrando el mouse será posible desplazar los campos.

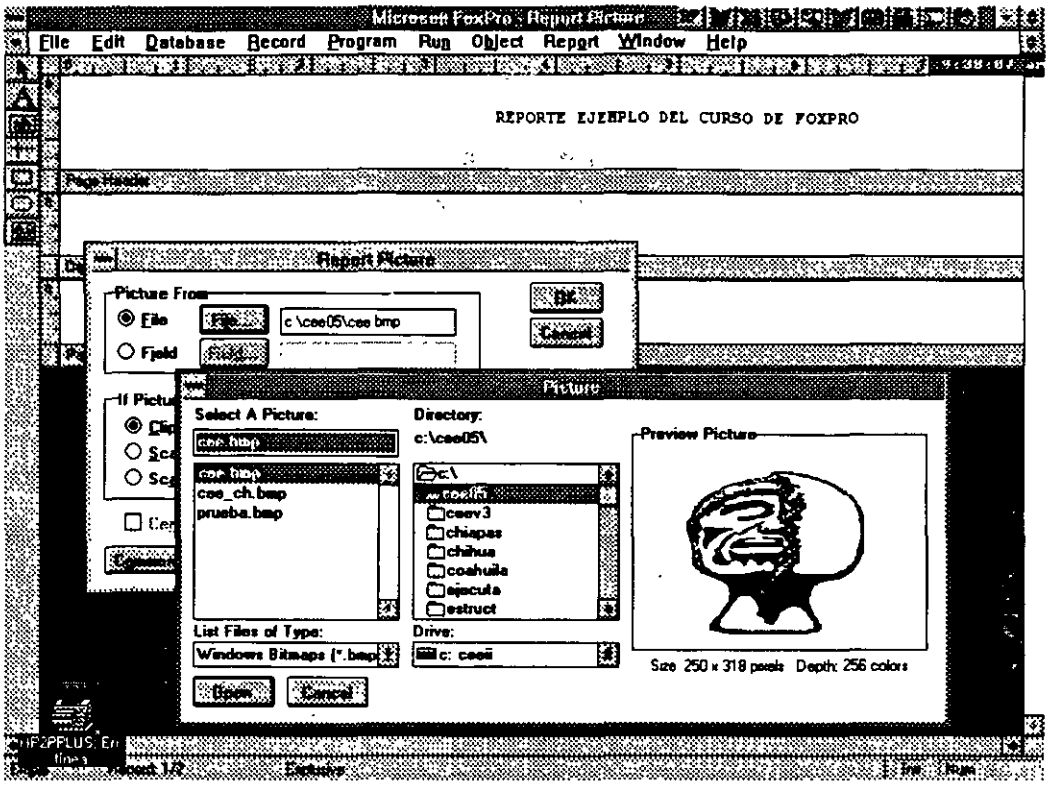


10.3.3 Dibujo y líneas

Los tres botones con líneas, cuadro y rectángulo redondeado nos define dibujos que se podrán incluir en los reportes. De la misma manera que en los botones anteriores, se deberá de seleccionar primero el botón y después en el área del reporte marcar donde se desea que aparezca.

10.3.4 Imágenes

Para añadir imágenes dentro de un reporte, se deberá seleccionar el botón de imagen y seleccionar el área del reporte donde se desea incrustar la imagen. Aparecerá un cuadro de diálogo donde aparecerá un botón File que nos servirá para seleccionar el archivo que se desea incluir.



10.4 Partes del reporte

Definiremos como partes principales del reporte 3 el encabezado, el cuerpo del reporte y el pie de página del reporte. Estos se podrán identificar dentro del reporte porque aparecen tres barras grises membretadas con los nombres correspondientes.

- **Encabezado**: En el área del encabezado se incluirá todo texto que se desee que aparezca al principio de cada hoja.
- **Detalle**: Esta región servirá para incluir todos aquellos campos o variables que aparecerán en todo el reporte.
- **Pie de página**: Aquí se definirá las variables o notas que se desea que aparezcan en el pie de todas las páginas.

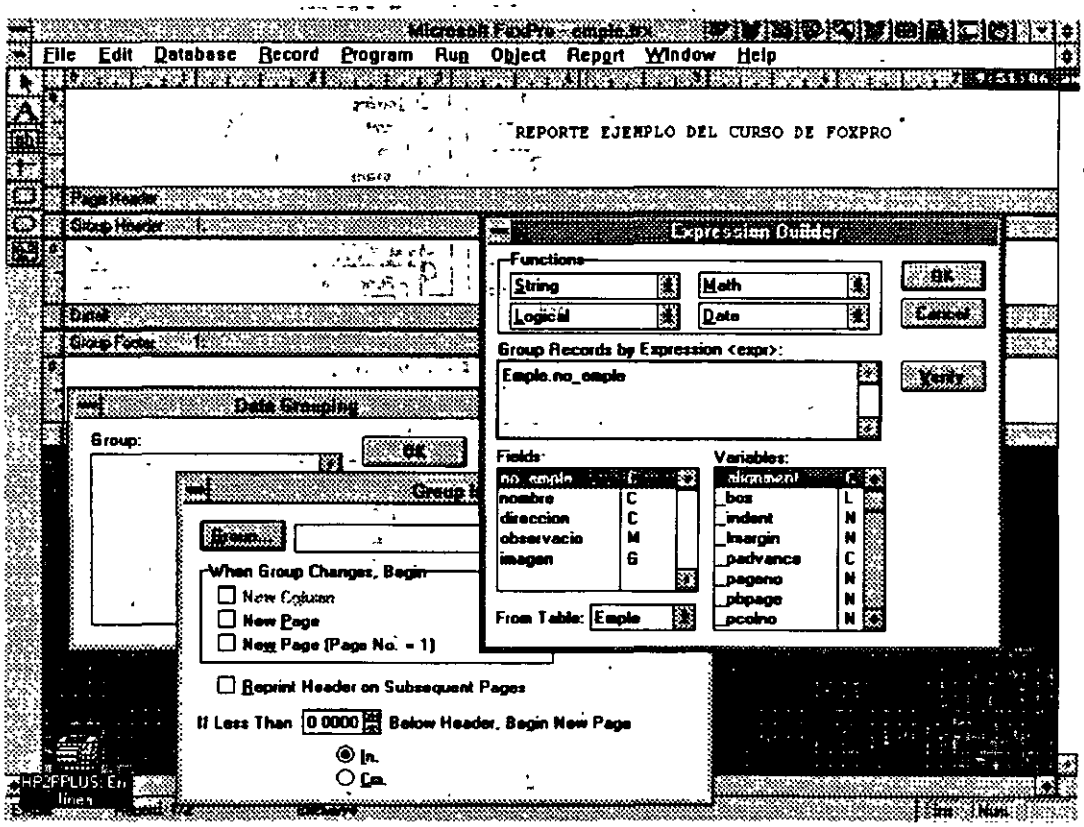
Estas barras será posible desplazarlas para poseer una mayor área de escritura por medio de los botones laterales que se encuentran al inicio de cada barra que delimita el área de trabajo. Mas adelante veremos que es posible que aparezcan dos barras mas que definirán los encabezados de grupo.

10.5 Creación de grupos de trabajo

Para crear grupos de trabajo se deberá llamar al menú desplegable Report-Data Grouping. Aparecerá la primera caja de diálogo donde se nos pregunta si deseamos añadir, borrar o crear un nuevo grupo. Si no tenemos ninguno debemos crearlo.

Si se seleccionar crear aparecerá una nueva caja de diálogo en donde se nos preguntará por el grupo, si oprimimos este botón aparecerá la siguiente pantalla en donde se nos desplegará los campos y variables existentes, los cuáles podremos seleccionar. Es necesario definirle al reporteador si cada grupo deberá de aparecer en la misma o en hoja por separado.

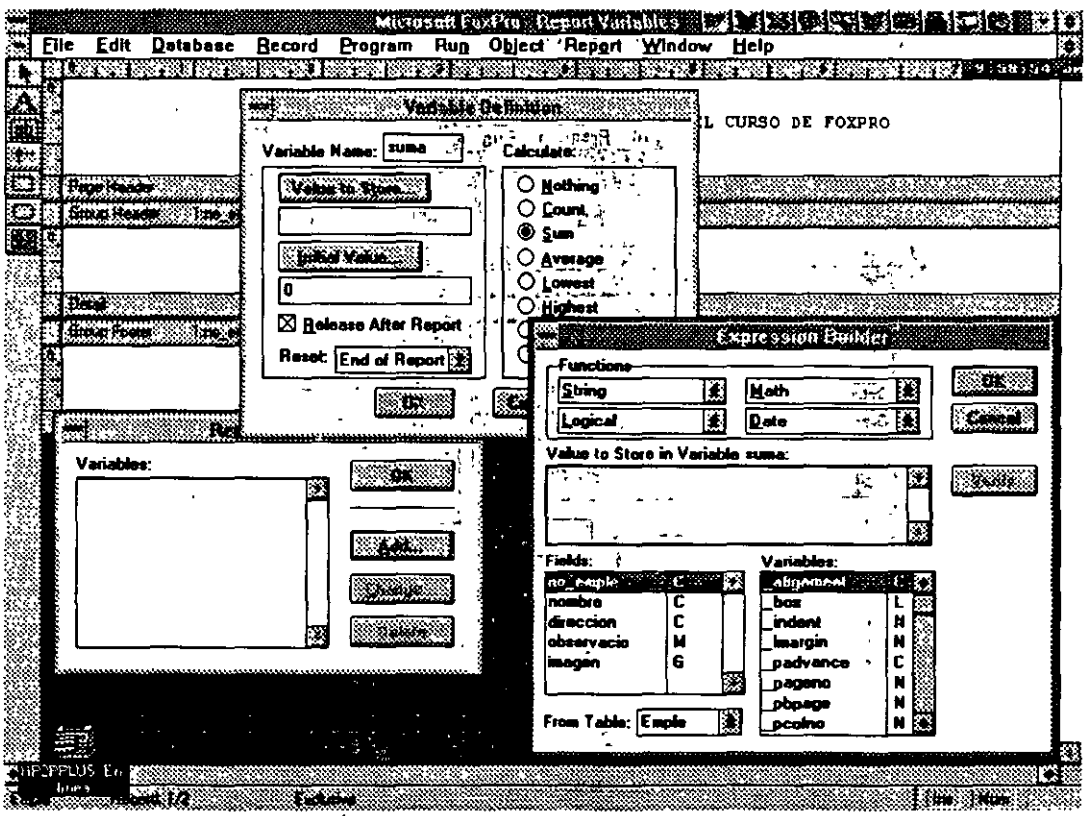
Nota: Para poder trabajar con agrupamientos será necesario abrir la base en su forma indexada utilizando como llave el campo que definiremos como campo de agrupamiento.



10.6 Creación de variables de trabajo.

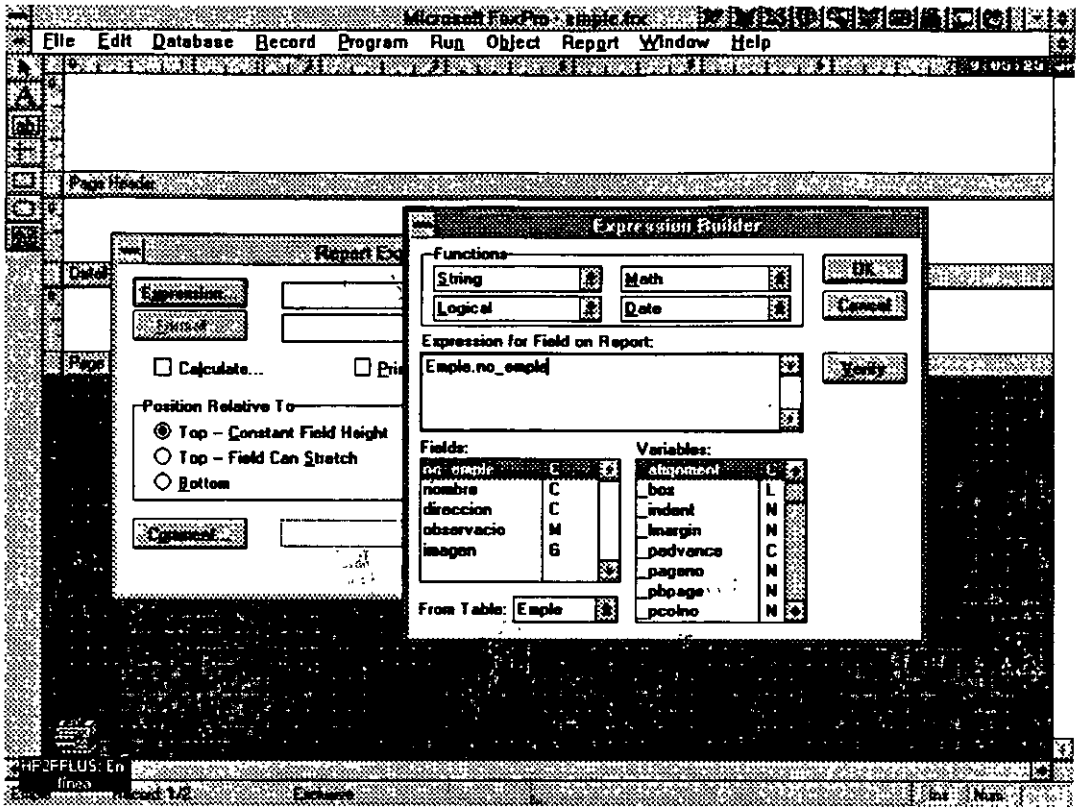
Las variables de trabajo serán campos que el usuario definirá, los cuáles almacenarán la información que el usuario requiera, su naturaleza es meramente matemática por lo que almacenará cálculos de variables numéricas.

Se seleccionará del menú Report-Variable, apareciendo una caja de diálogo en la que se nos solicita se indique cuál variable se desea borrar, modificar o crear una nueva. Crearemos una nueva, la nueva pantalla nos preguntará la función aritmética que realizará, se seleccionará una de ellas, el valor de inicio, su nombre y cada cuando se reinicializará. El botón de valor a grabar le indicaremos sobre que campo deberá de operar, para ello aparecerá la ventana con los campos de la base de datos, en donde seleccionaremos el deseado.



Una vez que se ha hecho toda le definición se deberá de añadir al reporte la variable creada, esto se hará en mediante el botón de cargar campos, sólo que en este caso la variable definida aparecerá

en la lista del lado izquierdo (Variables). El procedimiento será el mismo que en caso definido anteriormente.



10.7 Ejecución del reporte

Para ejecutar el reporte una vez creado, se deberá de salvar y ejecutar mediante la instrucción

REPORT FORM <NOMBRE REPORTE> <TIPO DE SALIDA>

TIPO DE SALIDA: TO SCREEN, TO PRINT, TO FILE