Te
UVG
Comp.
A 421
1990
c.1

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ciencias y Humanidades

IMPLEMENTACION DE UN GENERADOR DE REPORTES BASADO EN UN SISTEMA EXPERTO

AUGUSTO ENRIQUE ALONSO TELLO

BIBLIOTECA DE LA UNIVERSIDAD DEL VALLE DE GUATEMALA

Guatemala

1990

IMPLEMENTACION DE UN GENERADOR DE REPORTES

BASADO EN UN SISTEMA EXPERTO

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ciencias y Humanidades

IMPLEMENTACION DE UN GENERADOR DE REPORTES BASADO EN UN SISTEMA EXPERTO

AUGUSTO ENRIQUE ALONSO TELLO

Modelo de trabajo profesional presentado para optar el grado académico de Ingeniero en Ciencias de la Computación

Guatemala

1990

Vo. Bo.:

(f)_____
Licenciado David Alvarez
Asesor

Tribunal:

(f)_____
Licenciado David Alvarez

(f)_____
Licenciado Douglas Barrios

(f)_____
Ingeniero Luis Furlán

Fecha de aprobación: 30 de octubre de 1990.

A Dios.
A mis papás, por haberme dado la oportunidad de estudiar.
A mi esposa y a mi hija.
A mis amigos.

CONTENIDO

LISTA DE PANTALLAS

PREFACIO

En vista del auge que actualmente tiene el campo de la inteligencia artificial, surgió la inquietud de investigar técnicas nuevas como lo son los sistemas expertos. Por medio de ellos se pueden hacer aplicaciones de diferente naturaleza, una de las cuales, el generador de reportes fue analizado e implementado en el presente trabajo.

En el segundo capítulo de este trabajo se analizan los fundamentos teóricos de los sistemas expertos, enfocándose, principalmente, a sistemas basados en reglas. En el capítulo número tres se explican los pasos y lineamientos utilizados en el diseño del proyecto. Después de llegar a tener un prototipo completo, se analizan los detalles técnicos de la implementación de este proyecto en el capítulo número cuatro. Finalmente, se concluye acerca de todo el trabajo en el capítulo número cinco.

I. INTRODUCCION

El presente trabajo tiene por objeto realizar un generador de reportes basado en un sistema experto. El campo de la inteligencia artificial se ha puesto de moda en el medio de la informática para la resolución de problemas. Una de sus ramas, los sistemas expertos, son mecanismos que permiten resolver problemas, ya que poseen bancos de conocimientos acerca del problema dado. El tamaño del conocimiento depende del problema a resolver.

El aporte hecho pretende poner al alcance de usuarios de poca experiencia o sin conocimiento alguno de programación, una herramienta que le permita elaborar reportes para un lenguaje de cuarta generación, de gran demanda en el medio guatemalteco, actualmente. Esto se hizo proporcionando, inicialmente, una breve descripción de los principios básicos que tiene un sistema experto, en especial sistemas basados en reglas. Se presenta, además, una explicación de lo que son los generadores, sus componentes y la aplicación de los mismos.

Después de tener la base teórica necesaria, se muestra la forma como se diseño el generador. Para el efecto se presentan los lineamientos utilizados para que el desarrollo fuera dinámico. Se da una explicación de los diferentes módulos del generador, los algoritmos utilizados y las estructuras de

datos que lo conforman.

Debido a que se utilizó la técnica de prototipos para el diseño, la implementación fue bastante dinámica, ya que conforme se desarrollaba, se iba pensando la forma óptima como se diseñaría. En el desarrollo del proyecto se hicieron trabajos previos, para ver si realmente era posible implementar el generador basado en reglas. Después de tener resultados positivos, se implementó utilizando un lenguaje de programación que permitía el uso de estructuras de datos bastante flexibles.

Finalmente, se muestra la forma de utilizar el generador de reportes y se presentan las conclusiones que se obtuvieron al finalizar el presente proyecto.

II. SISTEMAS EXPERTOS

A. Principios generales y Características

La fase de la revolución en computación, que empezó a investigar sistemas expertos, data a principios de los años setenta. Mientras los especialistas en periféricos de computadoras estaban desarrollando tecnología de microchips, especialistas en programas - la gente que diseña y construye los programas que controlan las computadoras - querían darle un nuevo viraje al desarrollo. Este viraje no fue la invención de una nueva forma de codificar información, sino que fue un cambio conceptual en las ciencias de la computación, conocido como Inteligencia Artificial.

La meta de los científicos de inteligencia artificial ha sido siempre desarrollar programas de computadoras, que en cierto grado puedan pensar. Esto es resolver problemas en una forma que pueda considerarse inteligente, si fuera hecho por humanos. Sistemas expertos es el producto de 20 años de investigación, para definir la apropiada naturaleza de estos programas.

En los años sesentas, los científicos en inteligencia artificial trataron de simular procesos complicados de razonar, al buscar métodos generales para resolver problemas típicos de propósito general. Desarrollar programas generales

fue bastante difícil y sin resultados significativos. En vista de ello, decidieron que debería de existir una forma de hacer los programas de computación inteligentes. Era bastante difícil hacer un programa de propósito general, por lo que se concentraron en métodos y técnicas generales a utilizar en programas más especializados. Durante los setentas, se investigaron técnicas como la Representación - formular un problema para que sea fácil de resolver - y Búsquedas - como controlar búsquedas para una solución, de manera que no use mucha memoria de la computadora ni se tarde tanto. De nuevo los resultados no fueron tan satisfactorios.

Fue a finales de los setentas, cuando los investigadores en inteligencia artificial empezaron a descubrir algo importante: Que el poder de la solución de un problema proviene del conocimiento que posee, no necesariamente de las inferencias y formalismos que utilice. El viraje conceptual estaba hecho según Buchanan (1984:30) y se resumiría en el siguiente párrafo:

> "Para hacer un programa inteligente, hay que proveerlo de cantidades de conocimiento específico de alta calidad sobre un problema."

Esto permitió el desarrollo de programas de computación de propósito especial, sistemas que eran expertos en alguna área de un problema. Estos programas fueron llamados Sistemas Expertos, y así empezó una nueva área.

A pesar que el campo de sistemas expertos es relativamente nuevo, existe ya una formal definición de éstos. Un sistema experto se define como un programa inteligente que utiliza conocimiento de un experto humano y procedimientos de inferencia para resolver problemas que son suficientemente complicados como para requerir la experiencia humana para su solución.

El objetivo más importante para un sistema experto es lograr el nivel de funcionamiento que un experto llegaría a alcanzar. En este caso, siginifica producir resultados de calidad en un tiempo mínimo.

Los sistemas expertos, a diferencia de la programación convencional, son altamente interactivos. Un usuario, por ejemplo, puede detener el procesamiento de un sistema experto en cualquier momento y preguntar al sistema experto el "por qué" de muchas de sus acciones o conclusiones. El trabajo que hace un sistema experto ha sido hecho anteriormente por un especialista en un campo determinado.

El proceso de constriuir un sistema experto es llamado ingenieria de conocimiento. Tipicamente involucra una forma especial de interacción entre un constructor del sistema experto, llamado Ingeniero de Conocimiento, y uno o más expertos en algún problema. El ingeniero de conocimiento

extrae, de los demás expertos, sus procedimientos, estrategias, y reglas para resolver el problema, y construye el sistema experto basado en este conocimiento.

El resultado es un programa de computación que resuelve problemas de manera similar como expertos humanos. El siguiente enunciado de Paul E. Johnson, un científico que se ha dedicado durante años a estudiar el comportamiento de expertos humanos y describe el significado del término "experto" así (1985:54) :

> "Un experto es una persona que debido al entrenamiento y experiencia, es capaz de hacer cosas, que el resto de la gente no puede".

La noción central de resolver problemas, inteligentemente, radica en que el sistema debe construir la solución en forma selectiva y eficiente de un conjunto de alternativas. Un sistema experto posee un desempeño óptimo al utilizar el conocimiento en el menor tiempo. El corazón del sistema experto es el cuerpo del conocimiento que acumula durante su construcción. El conocimiento es explícito y organizado para simplificar la toma de decisión. La importancia de esta característica se tiene con la acumulación y codificación del conocimiento. Otra característica de gran utilidad para el sistema experto es el nivel elevado de experiencia que posee para resolver problemas. Esta experiencia puede representar la mejor forma de pensar dentro de los expertos en un campo,

dejando así la solución a un problema exacta y eficiente. La elevada experiencia junto con la habilidad aplicada permite al sistema un costo eficiente, capaz de crecer por sus propios medios en el mercado comercial. La flexibilidad del sistema también ayuda aquí, ya que puede crecer paulatinamente para encontrar las necesidades del negocio o institución. Esto significa que se puede empezar con una inversión modesta y expandirse según las necesidades lo demanden.

El poder predecible de modelar del sistema experto le permite que actúe como un modelo o teoría de procesar información, para resolver problemas en un dominio dado, proveyendo las respuestas deseadas para un problema específico y mostrando los cambios que se tendría para nuevas situaciones. El sistema experto puede explicar en detalle cómo la nueva situación apunta al cambio. Esto permite que el usuario evalúe los efectos potenciales de nuevos hechos o datos y entender las relaciones para la solución. Similarmente, el usuario puede evaluar los efectos de nuevas estrategias o procedimientos en la solución, al agregar nuevas reglas o modificar las existentes.

El cuerpo del conocimiento que define la destreza del sistema experto, también provee una característica adicional, una memoria institucional. Si la base del conocimiento es

desarrollada a través de iteracciones por una llave desde una oficina, departamento, o sección, representa la política actual o el procedimiento operacional de ese grupo.

Por último, un sistema experto tiene la habilidad de proveer una facilidad de entrenamiento para personal clave y miembros importantes. Sistemas expertos pueden ser diseñados para dar este entrenamiento, desde que contienen el conocimiento necesario y la habilidad para explicar el razonamiento.

B. Sistemas basados en reglas

Hemos mencionado que el corazón del sistema experto es el cuerpo del conocimiento estructurado para soportar la toma de decisiones. Como está organizado y representado este conocimiento ?

La forma como el conocimiento está estructurado en un programa es la representación del conocimiento. La representación del conocimiento es el área más importante de la inteligencia artificial. Para resolver problemas, analizar imágenes, procesar lenguajes naturales, etc., antes, un sistema inteligente tiene que poseer la capacidad de poder representar todo ese conocimiento que es la base de su inteligencia. Hay un conjunto de estándares para la

representación, como lo son las reglas, las redes semánticas, objeto-atributo-valor, los marcos y las expresiones lógicas. En esta tésis se discutirá en detalle las reglas, que es la que se usará para la construcción del sistema experto.

La representación del conocimiento basado en reglas, se apoya en el uso de condiciones (IF) y acciones (THEN). Una regla es el elemento dentro de la base del conocimiento que provee una forma flexible de representar el conocimiento. Con la regla se puede afectar la forma en que los valores de los parámetros se infieren o la forma en que se comunican al usuario. Cuando la situación de un problema satisface o cumple la parte de la condición de la regla (IF), la acción especificada por la regla se lleva a cabo (THEN). Cuando esto ocurre se dice que la regla se ejecutó, en el caso contrario se dice que la regla falló. Las reglas se usan para capturar la clase de respuesta semi-lógica a patrones familiares que caracterizan mucho el pensamiento humano diario. Un intérprete de reglas compara las condiciones de las reglas con los hechos y ejecuta la regla cuya condición coincida con el hecho que se desea solucionar, o sea que actúa como un mecanismo general de razonamiento. La acción de las reglas puede modificar el conjunto de hechos en la base del conocimiento, por ejemplo, agregando nuevos hechos. Estos nuevos hechos agregados a la base del conocimiento pueden

usarse para formar concordancias con la parte de la condición de la regla (IF).

La acción de la regla puede afectar el mundo exterior, dirigir el control de un programa o dar una orden al sistema para que busque una solución. Las reglas proporcionan una forma natural de describir procesos orientados hacia un complejo y rápido cambio del ambiente. Un conjunto de reglas puede especificar la manera de representar recomendaciones, directivas o estrategias; son a menudo apropiadas cuando el dominio del conocimiento resulta de asociaciones empíricas desarrolladas a través de años de experiencia resolviendo problemas en un área.

Al tener el conocimiento codificado en reglas, tenemos lo que se conoce como un Sistema de Producción. Si el sistema de producción se utiliza para construir sistemas basados en conocimiento de propósito específico, se constituye un sistema experto. En un sistema de producción se ven involucrados tres componentes:

- Una base de reglas, formada por el conjunto de reglas de producción.
- El contexto que es una base de datos global.
- Un intérprete que controla el comportamiento del sistema.

A continuación se presenta un ejemplo de un sistema de producción:

P1

IF (condición No. 1)

THEN (acción No. 1)

P2

IF (condición No. 2)

THEN (acción No. 2)

.
.
.
.
.

Pn

IF (condición No. n)

THEN (acción No. n)

El cumplimiento de parte de la condición de una regla hacia los hechos produce lo que se conoce como "cadena de inferencia". La cadena de inferencia se forma de sucesivas ejecuciones de reglas, e indica la forma como el sistema utiliza las reglas para inferir una solución a un problema dado. En un sistema experto, la cadena de inferencias puede ser desplegada al usuario para explicarle la forma como el sistema llegó a la conclusión.

Existen dos formas importantes de como las reglas pueden ser usadas en un sistema basado en reglas; una es llamada Encadenamiento hacia adelante, y la otra Encadenamiento hacia atrás. En el encadenamiento hacia adelante la búsqueda de nueva información va partiendo de varias reglas hacia lo que se desee probar. Se aplica la regla, actualizándose la base de datos y la búsqueda se reinicia. Este proceso continúa hasta que se llega a la solución o no se encuentran reglas aplicables. Para el encadenamiento hacia atrás, se empieza con lo que se quiere probar, y únicamente se ejecutan las reglas relevantes a establecerlo. Si las condiciones de una regla igualan los hechos existentes en la base de datos, se aplica la regla y el problema está resuelto.

Si se tiene que dos o más reglas pueden ser válidas para determinados hechos, se debe contar con estrategias de resolución de conflictos. A continuación se mencionan algunas posibilidades:

- Especificar Ordenamiento: suponiendo que las condiciones de una regla que se ejecuta es un subconjunto de las condiciones de otra regla que se ejecuta. Utilice la regla con el subconjunto en la base que sea más especializada para la actual situación.
- Ordenar Reglas: arreglar todas las reglas en una

lista por prioridad. La regla que sea ejecutada antes en la lista tendrá la prioridad más elevada. Las otras serán ignoradas.

- Ordenar Información: arreglar todos los aspectos posibles en una lista por prioridad. La regla que sea ejecutada con la condición de prioridad más elevada tendrá la prioridad más alta.
- Ordenar Tamaño: asignar la prioridad más alta para aquella regla que tenga los requerimientos de constricción elevados.
- Limitar Contexto: reducir el conflicto por similitud, separando las reglas en grupos, algunos de los cuales estarán activos en cualquier momento. Tener procedimientos para activar y desactivar grupos.

C. Generadores usando sistemas expertos

Un generador es una técnica para resolver un problema al producir posibles soluciones, las cuales serán aceptadas o rechazadas después que hayan sido evaluadas. Dependiendo del propósito y la naturaleza del problema, el generador puede generar todas las posibles soluciones, antes que sean evaluadas, o generar y evaluar al mismo tiempo. La acción terminará cuando una solución aceptable sea localizada, que

un número de soluciones se complete, o que todas las soluciones sean generadas y evaluadas. Un generador toma como entrada de datos las especificaciones del usuario y las traduce a un programa que realice tales especificaciones.

Entre las características que debe contar un buen generador, se mencionan las siguientes:

- Completo: producir eventualmente todas las posibles soluciones.
- No redundante: no perjudicar la eficiencia, al tener repetida la misma solución.
- Bien informado: se limitará la información, respecto de la solución que se haya propuesto.

Un generador que se basa en un sistema experto tendrá las diferentes soluciones, en este caso el conocimiento, en una representación permitida por sistemas expertos, como son las reglas. Debido a que nuevas reglas pueden ser agregadas para nuevas situaciones, sin afectar las existentes, el generador puede ir creciendo y ser bastante flexible, ya que bastará agregar nuevas reglas o cambiar condiciones y acciones según la necesidad.

III. DISEÑO DEL PROYECTO

A. Lineamientos utilizados

En el diseño general del proyecto se siguieron ciertos lineamientos para que el desarrollo del mismo fuera dinámico y con posibilidades a cambios futuros.

1. Modular

Los diferentes componentes que conformaban el proyecto fueron formando módulos, de acuerdo a la naturaleza de los mismos. De esa forma se tenían módulos para todo lo referente a las reglas, otros para las funciones y procedimientos de entrada de datos y verificación, manejo de ventanas y menús, etc. Esto se logró gracias a las ventajas que proporcionó el lenguaje de programación Turbo Pascal 5.0 con las llamadas "unidades" (UNIT). Cada una de ellas tiene sus propias funciones, procedimientos y estructuras, y pueden ser utilizadas por todo el sistema.

2. Estructurado

Se tomaron en cuenta los principios de una programación estructurada, como es el uso de funciones y procedimientos, nombres de variables consistentes, manejo de mensajes de error, etc. Asimismo, se trató de hacer

compacto a cada procedimiento, de manera que fuera lo más general posible.

3. Eficiente

La forma de escribir futuros programas o rutinas se dejó de una manera sencilla, al igual que el mantenimiento. El tiempo de corrida es aceptable y los resultados obtenidos son los esperados por el usuario ante el problema planteado.

4. Amigable con el usuario

La interacción con el usuario se estableció a través de menús y ventanas. Utilizando las flechas para desplazarse por las diferentes opciones , y para escoger la opción deseada con la tecla de retorno. Para cerrar una ventana se debe oprimir la tecla "DEL", y para salir del sistema con la tecla "ESC". El objetivo de hacer la interacción por menús y ventanas es para dar la oportunidad a usuarios con poca experiencia, y hacer amigable su utilización, ya que los comandos o teclas que el usuario debe utilizar son estándares para las diferentes partes del sistema.

5. Utilización de prototipos (Prototyping)

El uso de prototipos para el diseño fué determinante.

En el nivel más básico, el proceso de prototipos puede describirse como sigue: Se define una idea y se descompone a su forma más simple; los modelos en escala se construyen e interactúan de una manera bastante significativa; los modelos se cambian y refinan. La secuencia se repite hasta que todo el diseño esté completo. El modelo en escala debe sacrificar algunos aspectos del sistema final, mientras preserva otros, a fin de ser construido más rápidamente. De esa manera, cualquier aspecto que no haya sido modelado puede ser eliminado.

Un sistema puede tener muchos prototipos que construidos simultáneamente, cada uno modela diferentes aspectos del producto final. Estos aspectos son aquellos en los cuales existe mucha incertidumbre o son lo más vital para el éxito de la realización del proyecto.

B. El generador de reportes como el problema a solucionar

Un generador de reportes es una herramienta que nos permite obtener un reporte después que el usuario le ha proporcionado las especificaciones que desea. Este generador se basó en la sintaxis de ACE, generador de reportes para Informix-Sql. Se escogió esta sintaxis debido al auge que tiene actualmente el Informix en el área de programación, asi

como la especificación del reporte en sí, la cual es bastante entendible y compacta. Estas especificaciones consisten en diferentes secciones, las cuales se describen a continuación:

- DATABASE Section: en esta sección se identifica la base de datos que el reporte utilizará; es necesario que todo reporte la posea.

- DEFINE Section: esta sección es opcional y se utiliza para declarar variables usadas en el reporte y parámetros que el reporte pueda aceptar desde comandos en línea.

- INPUT Section: sirve para pasar parámetros al reporte, es opcional.

- OUTPUT Section: en esta sección se controla el largo de la página, el tamaño de los márgenes y el control del direccionamiento de la salida de un reporte.

- SELECT Section: especifica las columnas y tablas que se utilizarán, así como condiciones que se tomarán en cuenta, ordenamiento de la información y su agrupamiento por un campo dado. Esta sección es necesaria para el sistema.

- FORMAT Section: esta sección determina la forma como el reporte aparecerá.

El generador produce un reporte con todas las especificaciones requeridas de las secciones anteriores, el cual funciona al compilarse con Informix.

C. Lógica y algoritmos del generador

Debido a que el generador fue hecho en forma modular, ya que se basó en prototipos, se explicarán en forma independiente cada uno de los módulos. Sin embargo, muchos de ellos interactúan y dependen entre sí.

1. Generación de un nuevo reporte

La generación de un reporte consistirá en elaborar un reporte de acuerdo a las especificaciones y características dadas por el usuario. Para obtener esta información se cuenta con el interfase de menús y ventanas, por medio del cual, el usuario escogerá las diferentes opciones que pueden tener las secciones que conforman el reporte en sí. Para explicar la lógica y los algoritmos seguidos en la generación, primeramente se describirá el algoritmo global que involucra una generación, sin entrar en algún detalle. Seguidamente, cada paso que conforma el algoritmo global se explicará detalladamente.

El algoritmo global de una generación completa se presenta a continuación:

1. Desde el menú principal, seleccionar "Reporte".

2. Al tener las opciones del "Reporte", seleccionar "Generación.

3. Especificar el archivo que contiene la definición de la base de datos, las tablas y los campos.

4. Escribir el nombre que se desee lleve el programa para el reporte y el nombre de la base de datos.

5. Escoger las secciones que se desea contenga el reporte. Las secciones deben ser escogidas en el orden que aparecen, es decir de arriba hacia abajo, omitiendo la que se desee no aparezca en el reporte.

6. Para cada sección que se seleccionó, escoger las características que contendrá el reporte. Estas características serán interpretadas por el sistema experto compuesto por reglas, las cuales generarán el programa para el reporte especificado.

Antes de explicar las pasos de una generación, se describirá la secuencia que sigue el programa principal. Primero que todo, en el programa principal se llama a la rutina "Initialize", la cual inicializa variables globales, define atributos para la pantalla y habilita el archivo que contiene las reglas de nuestro sistema experto. Después se

llama a la rutina "Menu_Prin", la cual se encuentra en un ciclo, hasta que se oprima la tecla de "Esc", que significa salir del programa. En esta rutina se despliegan las opciones principales con las que cuenta el generador, como son Reporte, Reglas y Configuración. Cada una de las cuales representa un módulo del sistema, y fueron desarrolladas y diseñadas paralelamente utilizando prototyping (prototipos). Si se tecleó el "Esc", se cierra el archivo que contiene las reglas, se define una ventana de 80x25 y el programa termina.

Ahora se describirá la lógica y el algoritmo seguido para cada uno de los pasos anteriormente mencionados, de una forma más detallada.

1. La forma de escoger una opción en el menú principal, es por medio de las flechas del cursor (derecha -> e izquierda <-), posicionándose en la opción deseada, en este caso "Reporte" y oprima la tecla de retorno. Se utiliza la función "Readkey" de Turbo Pascal para que el programa sepa cuál tecla fue presionada. Ya teniendo identificada la tecla de retorno, se llama la rutina "Escoge_Opcion_Menu", a la que se le manda como parámetro el número de la opción que corresponde para "Reporte", en nuestro caso es uno. Esto activará la ventana que contiene las opciones del "Reporte", entre las cuales se encuentra la "Generación".

2. Para escoger la opción de "Generación" se posiciona el

cursor utilizando las flechas del cursor (arriba y abajo) y se oprime retorno.

3. Cada vez que se hace una generación de un reporte, es necesario tener las especificaciones del ambiente en que los datos se encuentran. Esto viene siendo el nombre de la base de datos, las tablas que conforman la base de datos, y los campos con sus respectivos tipos y longitudes que, a su vez, forman las tablas. Esta información se encuentra en un archivo de texto grabado en el disco. Para ingresarlo se activa una ventana, donde se le pide al usuario que ingrese el nombre del mismo. Si el nombre ingresado existe grabado en disco, se llama a la rutina "Procesa_Schema", la cual va leyendo el archivo por medio de un analizador de léxico (scanner). El analizador de léxico separa los diferentes tokens y llena una tabla con dicha información, la cual se clasifica según el tipo, es decir, si es la base de datos, una tabla, el nombre de campo, el tipo del campo y su longitud. Esta tabla es de gran utilidad, ya que le da al usuario el ambiente con el que puede generar el reporte, desde el punto de vista de la información.

4. Para el nombre del programa se activa una ventana, donde se pide que se ingrese el nombre que desee, siempre que tenga terminación ".Ace", para que al generar el programa y

se utilice el compilador de informix, este pueda leerlo sin problemas por las diferentes opciones. Por omisión, el nombre del archivo es "Nuevo.Ace". Ya teniendo el nombre, se crea en disco, un archivo con ese nombre, para lo cual se utiliza el "Rewrite" de pascal y se cierra la ventana.

El nombre de la base de datos se obtiene con la rutina "Database_Section", a la que se le manda el parámetro "G", que significa generación. Se activa una ventana y se llama la rutina "Crea_Inicio_Lista_G_M", por medio de la cual se crea el nodo cabeza de la lista encadenada que contendrá la información para que el sistema experto, nuestras reglas, generen el código correspondiente. En la ventana activa, el usuario puede seleccionar por medio de la tecla de retorno, el nombre de la base de datos que utilizará el reporte. Al seleccionar la base de datos, del lado izquierdo de la misma aparecerá un asterisco. Este se utiliza como marca para saber cual fué lo escogido, ya que, en un principio, ninguno tiene asterisco. Con el nombre establecido se llama a la rutina "Crear_Lista_G_M", que crea un nodo con la información del nombre, y se crean otros nodos necesarios para la sección Database. Ya teniendo la lista encadenada completa, por medio de la rutina "Reglas_G_M", se procesan todos los nodos en las diferentes reglas que se tienen para generar. Finalmente se borra la lista encadenada, con la sentencia "Release", para que pueda ser utilizada por otra sección.

5. Como se mencionó anteriormente, para la generación de las diferentes secciones del reporte, se debe seguir el orden de arriba hacia abajo que aparece en la ventana que se encuentra activa. Se insiste en este orden, ya que la generación del código de las secciones, debe tener un orden lógico, para que al momento de compilar con informix, no nos dé ningún problema.

Debido a que las secciones que generan código tienen una lógica similar a la que tiene la sección Database, es decir:

a) Crear inicio de lista encadenada.

b) Obtener especificaciones requeridas.

c) Crear un nodo para cada especificación.

d) Procesar la lista en las reglas del sistema.

e) Eliminar lista.

se mencionarán únicamente aspectos propios y relevantes de cada sección.

5.1 En la sección DEFINE, las variables que se definen son inicializadas con el nombre "variable", concatenado con un contador que se tiene el valor inicial de uno. Conforme se van definiendo nuevas variables, el contador se va incrementando para que tengan valores nuevos. Si el usuario desea, el nombre de las variables puede ser escogido por él. Este ingreso de variables se hace por medio de la rutina "EditStr", la cual tiene como parámetros la fila, la columna,

BIBLIOTECA
DE LA
UNIVERSIDAD DEL VALLE DE GUATEMALA

el largo de la variable, y el valor con el cual se inicia dicha variable. Al ingresar una variable, de inmediato se activa una ventana donde se encuentran los diferentes tipos permitidos para las variables. Utilizando el cursor se puede posicionar en el tipo deseado, y oprimiendo la tecla de retorno se escoge el mismo. Las variables que se definen son almacenadas en un arreglo, ya que servirán en la sección del INPUT.

5.2 Si el usuario desea tener ingreso de información en el reporte, se utiliza la sección del INPUT. Para el efecto se le presentan las variables que fueron definidas en la sección DEFINE, las cuales se escogen de la ventana que se encuentra activa. Se sigue el procedimiento similar de escoger explicado anteriormente. El texto que tendrá este ingreso, se inicia con "Texto de Ingreso", utilizando la rutina "Editstr", por lo que el usuario puede cambiar al texto que considere más conveniente. Se permite un máximo de 25 caracteres, los cuales pueden ser incrementados cambiando el parámetro de "Editstr", que se refiere al largo del texto a editar; se recomienda además incrementar el ancho de la definición de la ventana.

5.3 En la sección del OUTPUT, básicamente por medio del cursor iremos escogiendo las definiciones para el tamaño de los márgenes, largo de la página, direccionamiento a impresor

o archivo. La lógica será la misma para todas las opciones. Al tener la ventana activa, se posiciona el cursor en la opción deseada, y se presiona la tecla de retorno. Seguidamente se activa otra ventana, donde se encuentra el valor que toma la opción por omisión. Esto se hace a través de la rutina "Editstr".

5.4 En la sección del SELECT, se escogen los campos que formarán parte del reporte, las tablas donde se encuentran estos campos, las relaciones existentes entre las tablas, las condiciones que imperarán en el flujo de la información y el ordenamiento que se desea tenga la información por medio de uno o varios campos. Para escoger las opciones anteriores, se sigue la misma lógica de las secciones explicadas previamente. En el caso de los campos, éstos se leen de la tabla que se formó al cargar la base de datos. El nombre del campo se concatena con el nombre de la tabla que lo contiene, separados por un punto. Al definir el campo en la lista de los campos del SELECT, se forma un sinónimo del nombre del campo, formado por la primera letra de la tabla y las cuatro primeras letras del campo. Este sinónimo es el que se utilizará en las demás partes de esta sección que requierán el nombre del campo. Al igual que los campos, las tablas se toman del arreglo formado al cargar la base de datos. Las condiciones por tener diferentes opciones, activan una

ventana, para que el usuario pueda escoger el filtro que más se adapte a sus requerimientos. Cada opción de las condiciones, a su vez activan más ventanas, donde se captura la información que conformará la condición deseada. Finalmente, para el ordenamiento de la información, se habilita una ventana con los campos disponibles, el cual se selecciona y se toma el sinónimo del que hablamos anteriomente.

5.5 Por último la sección del FORMAT, tiene una forma diferente de obtener la información para el reporte, ya que como encierra el formato propio del reporte, es necesario tener un mecanismo que nos permita dibujar cómo se desea que aparezca el reporte. Para el efecto el algoritmo que rige a las diferentes opciones de la sección del FORMAT es igual para todas. Primero se activa una ventana que contiene las opciones propias del formato. Estas se encuentran en el orden lógico que deben llevar, de arriba hacia abajo. Al escoger una opción se llama a la rutina "Format_Blocks", la que activa una ventana donde después irá apareciendo como es que va quedando el reporte. Después se llama a la rutina "Editor_Formato", pasándole como parámetro el nombre de la opción escogida. Esta rutina es un editor de pantalla, el cual tiene el control del contenido de un archivo. Es decir, que dicho contenido puede modificarse, borrarse, agregársele

o bien únicamente verse. En el editor es donde el usuario va ir dibujando la forma como quiere que quede la parte del formato que esté analizando. En el editor se le presenta al usuario las filas y columnas relativas a la posición del cursor, para que pueda situar un campo o un título en la posición que en el reporte quiera que salga. Debido a que el fin del editor es primordialmente para dibujar el diseño completo del reporte en sí, cuenta con comandos sencillos como son Ctrl K D, el cual sirve para grabar el diseño, Ctrl K Q que sirve para salirse del diseño sin guardarlo. El diseño es guardado en un archivo en disco, con el nombre de la opción seleccionada y con la terminación ".FOR". Este archivo debe ser procesado por la rutina "Analiza_Edición", la cual lee línea por línea hasta que se termine el archivo y va separando los campos o datos del reporte. Para diferenciar un campo de un dato o título, se le antepone el carácter '#'. En el caso de los campos, se pide si se desea que tenga totales y/o máscara para el mismo. Cada vez que se termina un diseño completo de un reporte, los archivos que contienen las diferentes partes del formato, son borrados para que queden habilitados para un diseño futuro.

Como se explicó en un principio, la lógica de las diferentes secciones es la misma, por lo que únicamente se dio una explicación de la forma como se capturó la

información y cómo es que era procesada.

Ya teniendo el reporte generado, se cuenta con opciones para listar el programa producido, eliminar un reporte o mostra el directorio de todos los reportes que se han hecho. Para listar el programa que se generó, se lee el disco línea por línea y se va escribiendo en la pantalla para que el usuario pueda ir viendo lo que generó. Si se desea eliminar un reporte creado, se pide el nombre del reporte y por medio de la sentencia "Erase" de Turbo Pascal, se borra.

2. Modificación de un reporte existente

Para el proceso de modificación se requiere que el reporte ya exista en disco. La modificación de un reporte hace cambios a las secciones DATABASE, DEFINE, INPUT y para la sección de FORMAT, en donde únicamente se trata lo referente a las columnas, y separación óptima que debe de existir entre campos y titulos, al cambiar filas y columnas, es decir, si se quiere el reporte para otro tipo de papel. Los cambios a la sección SELECT, no fueron implementados a través del sistema experto, ya que resulta más sencillo utilizar un procesador de palabras; igual situación ocurre con los campos y títulos de la sección del FORMAT.

El algoritmo para efectuar la modificación del reporte, tiene pasos comunes con la generación, por lo que únicamente

se mencionarán, ya que fueron explicados anteriormente. Los pasos del algoritmo se explican a continuación:

1. Desde el menú principal, seleccionar "Reporte".

2. Al tener las opciones del "Reporte", seleccionar "Modificar un Reporte Existente".

3. Especificar el archivo que contiene la definición de la base de datos, las tablas y los campos.

4. Ingresar el nombre del reporte que se desea modificar. Este debe existir en disco, de lo contrario el generador da un mensaje de error y termina.

5. Utilizando un analizador de léxico (scanner), se irá separando los diferentes "Tokens" que forman el archivo que contiene al reporte. Para el efecto se lee token por token, y se van almacenando en una lista encadenada utilizando la rutina "Crear_Lista", donde se identifican aspectos relevantes como son: número del token, el token en sí, la sección a la que pertenece y la marca que se utiliza para saber si ya fue procesado o no por las reglas, se inicia. Esto se hace hasta que se encuentra el fin del archivo.

6. El usuario debe responder si desea o no hacer cambios a las secciones permitidas del reporte, y a las filas y columnas. Se usa la rutina "Modificacion_Reporte", en donde

se habilita una ventana y se le pregunta al usuario por el cambio o no de las secciones

7. En cada respuesta que es afirmativa, se llama la rutina correspondiente a la sección. Se utilizan las mismas rutinas que se usan para la generación, especificando solamente que se trata de modificación, con el parámetro igual a "M". Ya estando en la rutina de la sección, se le presenta al usuario como es que está actualmente el reporte, por medio de la rutina "Localiza_Seccion". Los datos actuales se despliegan para que el usuario decida en ese momento si cambia o no la información. Para las modificaciones de formato, se utiliza el sistema experto, y que determine los cambios que considere pertinentes. En este caso se utilizan reglas que se refieren a formatos en general, es decir se verifica si la información está centrada, justificada a la derecha o izquierda, justificación proporcional, etc. Para tener una separación óptima entre campos y encabezados, se toma en cuenta el espacio disponible y la longitud de los campos. Después que todos los cambios han sido efectuados, ya sea por el sistema experto o directamente por el usuario, se escribe el archivo a disco. Para que el sistema experto efectúe las condiciones y ejecute las acciones para los cambios, se utiliza la lista encadenada que se mencionó en un principio.

3. Manejo de reglas

En las reglas se encuentra toda la información del sistema experto para que efectúe la generación y modificación del reporte. Debido a que nuevas reglas pueden ir surgiendo para nuevas necesidades, es preciso que estas estén almacenadas en una estructura que permita hacer cambios o modificaciones en forma dinámica a la base del conocimiento. Ante está situación se eligió utilizar un manejador de archivos que fuera compatible con Turbo Pascal, este es el caso de "Btrieve". Al utilizar las funciones e interacciones que existen entre turbo y btrieve, se almacenó las reglas en un archivo de btrieve. De esa forma el usuario puede agregar, modificar, eliminar y consultar las reglas que desee. Las reglas están divididas en dos clases: reglas para generación y reglas para modificación. Para el efecto se tiene un índice del archivo usando como llave la clase de la regla y el número correlativo que la identifica. De esa forma, al efectuar una generación o modificación, el intérprete únicamente hará uso de las reglas que necesite y no tendrá que evaluar las de una clase que no le interese. Por medio de la rutina "Abra_Archivo_Btrieve", se abre y habilita el archivo que contiene las reglas al inicio del programa principal. Es necesario que el "Btrieve" esté activo antes de utilizar el generador. Para efectuar el mantenimiento de este archivo que contiene las reglas, existen las siguientes

rutinas:

- Add_Reglas: por medio de este procedimiento se agregan nuevas reglas a la base del conocimiento. Para el efecto se ingresa la clase de la regla, y el número correlativo. Luego se llama a la rutina "Ejecuta1", cuyos parámetros indican el acceso por la llave y el tipo de "GET". En este caso se utiliza un "GET EQUAL" para ingresar al archivo. Esta rutina verifica si la regla ya fue ingresada o no. En el caso que ya existiera, nos da un mensaje y no permite que se duplique. Si es una regla nueva se ingresa la descripción de la regla, condiciones y las acciones a tomar por medio de la rutina "Obtiene_Datos_Reglas", a variables globales. Ya teniendo la toda la información se llama la rutina "Inserta1", que pasa el valor de estas variables globales, a variables que son permitidas por el Btrieve (Packed Array of Char) y, finalmente, se llama la rutina que hace el ingreso al archivo Btrieve, usando la operación de INSERT.

Las rutinas que se muestran a continuación tiene una lógica similar a la Add_Reglas, por lo que únicamente se explicará su función principal.

- Update_Reglas: con esta rutina se modifican los campos que constituyen la regla, como son su descripción, condiciones y acciones. Para el interfase con Btrieve se utiliza la operación UPDATE.

- Delete_Reglas: este procedimiento elimina una regla del archivo, utilizando la función DELETE para Btrieve.

- Get_Reglas: con esta rutina se puede consultar una regla. Usa la rutina GET_EQUAL de Btrieve.

- Directorio_Reglas: este procedimiento nos presenta un reporte de las diferentes reglas existentes por clase. Se ingresa la clase para la cual se desea la consulta. En el reporte sale el número de la regla y su descripción.

Para incorporar las reglas al generador se utiliza la rutina "Arma_Reglas", la cual forma procedimientos de turbo pascal, para cada regla existente por clase. Además arma el procedimiento que tiene todas las reglas de una clase, para que se vayan analizando todas y se acepte la que cumple la condición. El algoritmo de esta rutina es como sigue:

1. Se crea en disco el archivo "REGLAS.PAS", por medio de Rewrite, el cual guardará todos los procedimientos que se generen. Para incorporarlo al generador, se tiene un "Include" de este archivo, por lo que cada vez que se hagan cambios a la base del conocimiento del sistema experto, es decir las reglas, se debe compilar de nuevo el generador.

2. Para las dos clases permitidas se efectúa lo siguiente: se recorre todo el archivo que contiene los registros de las diferentes reglas y se genera, primero, una clase con sus respectivos procedimiento-regla y el proceso

que maneja el análisis de las diferentes reglas de una clase. Para el efecto, el nombre del procedimiento se forma anteponiendo "R_", a lo que resulta de concatenar el número de la regla, "_" y la clase de la regla. A las condiciones se le antepone el "IF" y se les pospone "Then Begin". Luego viene las acciones, las cuales se escriben sin cambio alguno, poniendo al final el "End" de las condiciones y el "End" del procedimiento-regla. Por ejemplo la regla 001 de la clase "A", es decir para modificación del reporte, es como sigue:

```
Procedure R_001_A;
Begin
  If condiciones
  Then Begin
    acción;
      .
      .
      .
      .
    acción;
  End;
End;
```

Las condiciones y las acciones que se generan para cada regla, a su vez son funciones y procedimientos ya incorporados al generador. En cada acción se pueden poner varias funciones y procedimientos separados por punto y coma.

El procedimiento que se arma para recorrer las reglas de una clase, hasta que una de ellas sea aceptada, tiene la siguiente lógica:

```
Mientras Bandera_Reglas sea falsa
Begin
```

```
R_001_A; If Bandera_Reglas es verdadera Salga del Ciclo
.
.
.
R_099_A; If Bandera_Reglas es verdadera Salga del Ciclo
End;
```

4. **Menú de ventanas**

Debido a que toda la interacción con el usuario es a través de menús en ventanas, a continuación se explica la lógica de las principales rutinas.

- Do_Menu: esta rutina despliega las opciones que posee un menú e indica el número de la opción que fue seleccionada. Para el efecto se tiene un ciclo que pone en invertido la opción donde el cursor está ubicado, y lleva un control de las flechas arriba y abajo, para que el cursor no se salga de las opciones disponibles. El ciclo termina cuando se presiona la tecla de retorno, la cual escoge la opción donde esté situado el cursor, o con la tecla "DEL", que cierra la ventana.

- OpenWindow: este procedimiento crea una ventana utilizando el comando window de turbo pascal. Cada vez que se hace una nueva ventana, se crea un nodo de una lista encadenada, de esa forma una ventana puede llamar a otra y, al desabilitar la última, se activa la primera.

- CloseWindow: esta rutina cierra la ventana que se

encuentra activa, al eliminar el nodo de la lista encadenada que la contiene. Para estandarizar el generador, esta rutina se llama cada vez que se presiona la tecla "DEL".

D. Limitaciones

El generador de reportes está diseñado para cubrir su función principal, como es generar o modificar un reporte que pueda ser compilado por Informix. Sin embargo cuenta con ciertas limitaciones, entre las que podemos mencionar las siguientes:

- Debido a que el interfase utiliza el Btrieve para almacenar las reglas, se tiene limitantes en cuento a memoria, ya que el Btrieve, debe ser residente en memoria. Ante esta situación, la dimención de las estructuras y las listas encadenadas tienen un límite, ya que no tienen toda la memoria disponible. Esto repercute en el sentido que archivos muy grandes que se deseen modificar puede que a la mitad del proceso sean cancelados. El generador puede modificar archivos de tamaño promedio, y que son los que generalmente se utilizan.

- Las reglas pueden ser modificadas en forma dinámica, debido a que se encuentran almacenadas en un archivo. Sin embargo, para que estén en condición de ser operadas, es necesario que el generador sea compilado cada vez que se haga

algún cambio a ellas, para que el nuevo archivo de procedimiento-regla que se genera sea incluido.

- Algunas cláusulas de Informix no fueron implementadas, como es el caso de: Group by, Having group by, e Into group de la sección Select.

- El interfase con el usuario es bastante sencillo, y está diseñado para usuarios con poca experiencia en computación, sin embargo es necesario tener nociones de lo que implica un reporte en cuanto a relacionar tablas (JOINS), hacer las condiciones, aspectos propios de informix, etc.

- El generador se utiliza para reportes que funcionan en Informix. Si se deseara hacer un reporte para otro lenguaje, sería necesario que el interface capturara la información requerida, y las reglas del sistema experto estuvieran acordes al lenguaje. Actualmente, las reglas son totalmente flexibles, ya que bastará ingresarlas, pero en el el caso del interfase serían cambios más significativos.

IV. IMPLEMENTACION

A. Trabajo previo realizado

En el desarrollo de cualquier proyecto es necesario que el diseñador realice una labor investigativa, antes de llevarlo a cabo. Debido a ello, antes de implementar y diseñar el generador en sí, se realizaron diferentes trabajos. Estos van desde estudiar lo referente a sistemas basados en reglas, hasta el desarrollo de un generador que funcionara, pero que no tuviera la estructura y eficiencia necesaria.

En el caso del desarrollo del generador, se implementó uno utilizando el compilador "Clipper" con la base de datos "Dbase". Para el efecto, todas las estructuras necesarias eran archivos de Dbase, utilizando índices y rutinas de búsqueda del tipo secuencial. Se escogió el compilador "Clipper" debido a la función "&", la cual toma el valor hacia donde apunta una variable. Esto se utilizó para probar las condiciones y las acciones, de manera fácil y rápida.

En el caso de las reglas, antes de implementarlas, se escribió primero todos los requerimientos que involucra un reporte en informix. Se estudió la sintaxis propia que tiene un reporte de "ACE" y se estableció cuáles deberían ser las funciones y procedimientos a implementar para lograr que el sistema experto funcionara.

B. Desarrollo del proyecto

1. Prototipo del proyecto

Como se mencionó anteriormente, el uso de prototipos fue determinante en el desarrollo del proyecto. Cuando ya se tuvo la idea clara de las opciones y resultados del generador, se hizo un modelo donde estaban todas las opciones, de las cuales únicamente las más relevantes funcionaban. Poco a poco, cuando una opción producía el resultado deseado, se pasaba a trabajar otra. Si al momento de estar desarrollando un módulo no se avanzaba, se pasaba a otro, de manera que la implementación no se estancara. El hecho de visualizar como es que quedaría el generador, nos daba la oportunidad de realizar un diseño e implemtación, donde se tomaba en cuenta todos los aspectos e interrelaciones de los módulos. Los módulos que formaron el prototipo del proyecto son:

- Reporte: contiene todas las opciones que manejan la generación y modificación del reporte.
- Reglas: como su nombre lo indica, es el sistema de reglas, tanto a nivel de archivo, como de procedimiento-regla.
- Funciones y procedimientos de las acciones y condiciones: son todas las rutinas que se implementarón para que se evaluara las lista encadenadas que contienen los

requerimientos del usuario, al igual que rutinas que ejecutan las correspondientes acciones.

2. Lenguaje utilizado y métodos de programación

El lenguaje de programación utilizado es Turbo Pascal 5.0, debido a las grandes facilidades que posee para el manejo y definición de estructuras de datos eficientes. También se tomó en cuenta que el Turbo Pascal tiene un interfase con el manejador de archivos Btrieve, ya que este fue el se usó para el almacenamiento de las reglas. Otro factor importante que se destaca de turbo pascal, es la alternativa de definir estructuras, variables, funciones y procedimientos que conformen unidades de trabajo (UNITS), las cuales pueden ser utilizadas por todo el programa.

En cuanto a los métodos de programación utilizados, se siguió todos los lineamientos de una programación estructurada.

C. Utilizando el generador

En esta sección se presenta una sesión completa de un usuario utilizando las diferentes opciones del generador de reportes, para obtener un reporte. Se muestran las pantallas, y el resultado que producen ante los requerimientos y necesidades del usuario.

Al tener el cursor del sistema operativo, teclear "Tesisfin". Enseguida se presenta la pantalla correspondiente al menu principal:

```
Reporte              Reglas              Configuración


<F1>-Ayuda        -Cursor  <Enter>-Opcion  <Del>-Cierra ventana <Esc>-Salir
```

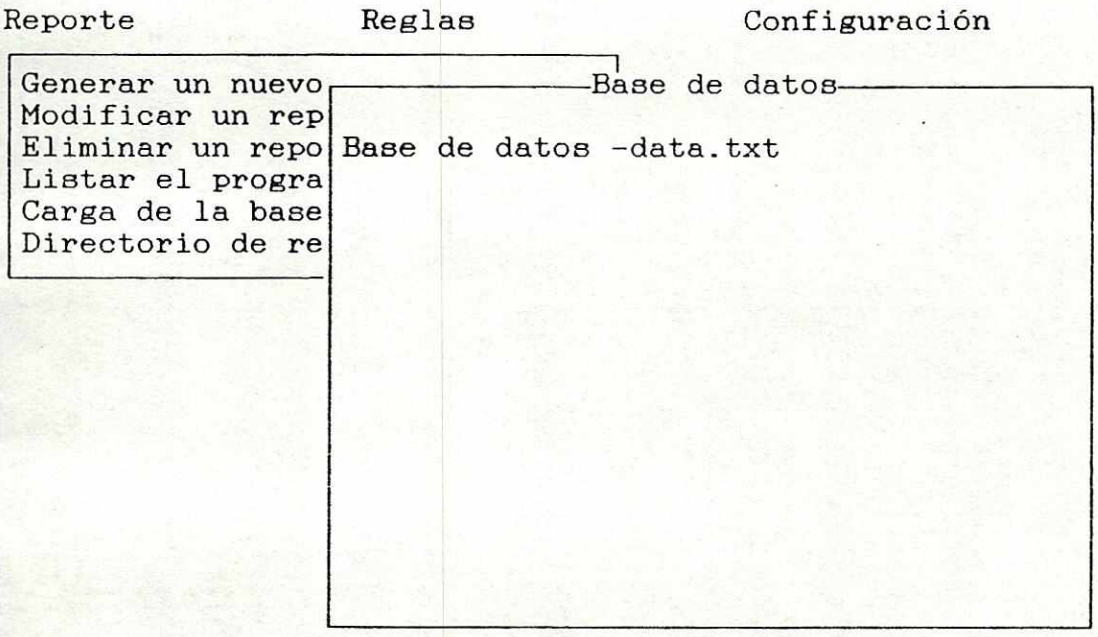
Pantalla No. 1

En el menú principal, por medio de las flechas del cursor nos posicionamos sobre "Reporte", y oprimimos la tecla de "enter", a continuación aparecerán las opciones correspondientes. Luego nos ubicamos en la opción para la generación y oprimimos "enter".

```
Reporte              Reglas              Configuración

┌──────────────────────────────────┐
│Generar un nuevo reporte          │ <=====
│Modificar un reporte existente    │
│Eliminar un reporte               │
│Listar el programa generado       │
│Carga de la base de datos         │
│Directorio de reportes            │
└──────────────────────────────────┘
```

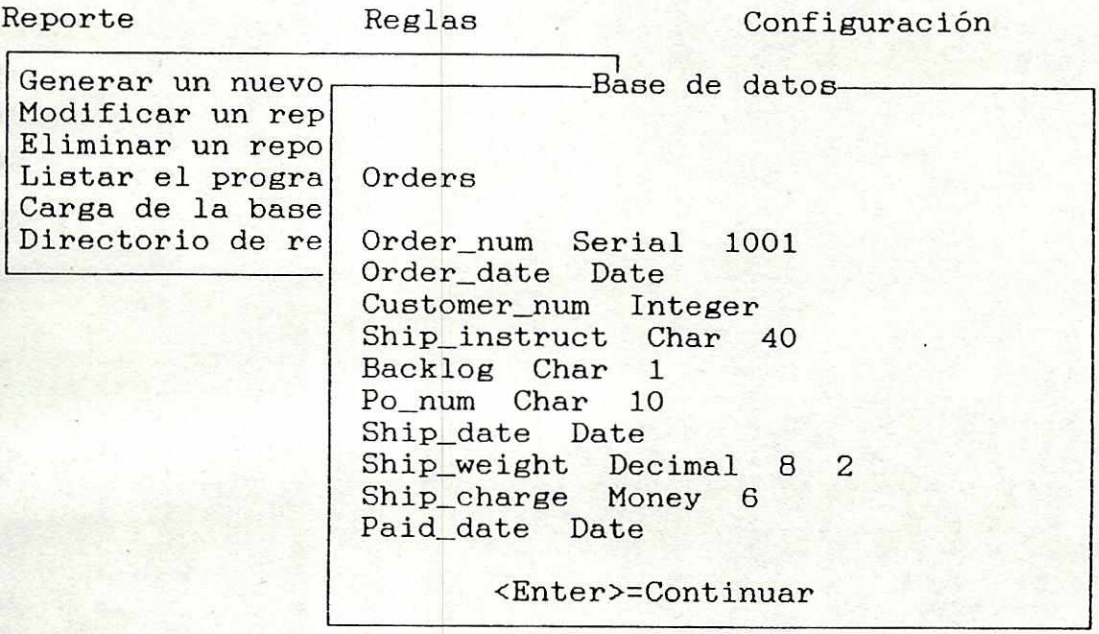
Pantalla No. 2

Al presionar enter sobre la opción de generación, se activa la ventana que nos pide que ingresemos el archivo donde está la definición de la base de datos, en es caso es: DATA.TXT.



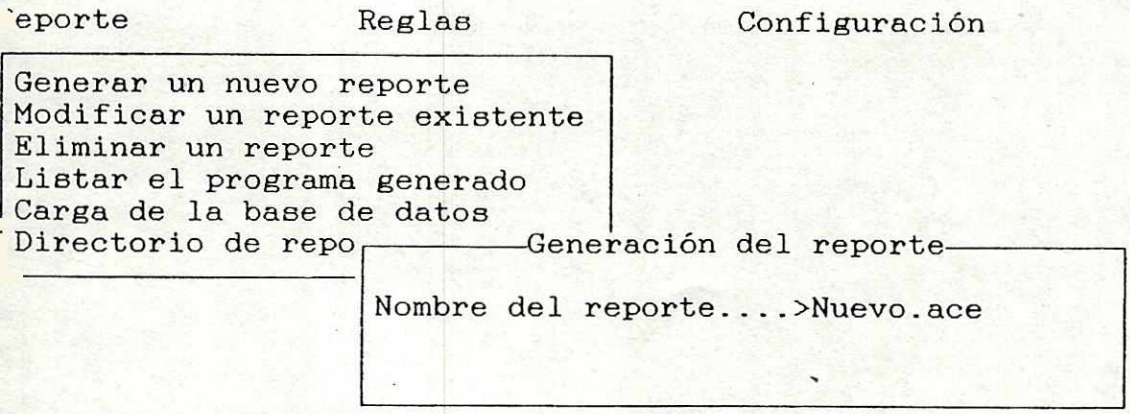
Pantalla No. 3

Al oprimir la tecla de "enter", después que se ingresó el nombre del archivo que contiene la definición, aparecerán los campos, los tipos de los nismos y tablas de la base de datos.



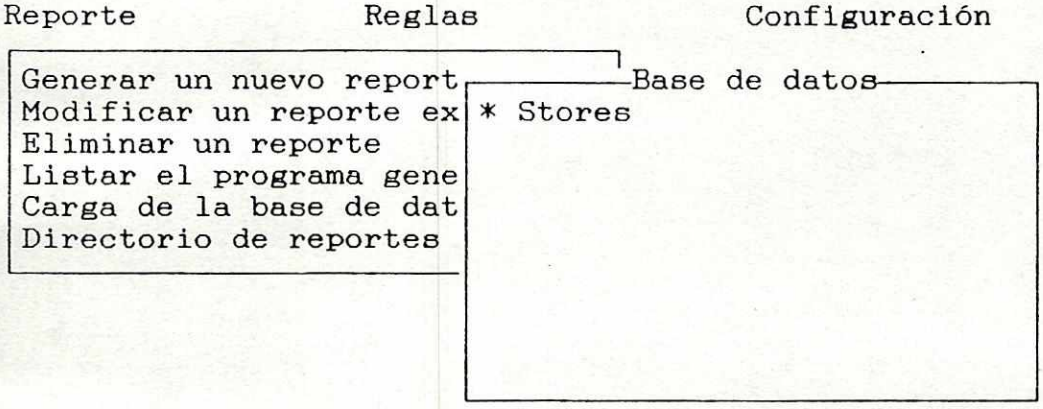
Pantalla No. 4

En seguida se activa la ventana, donde se ingresa el nombre del archivo en disco que contendrá el reporte generado. En este caso nuevo.ace.



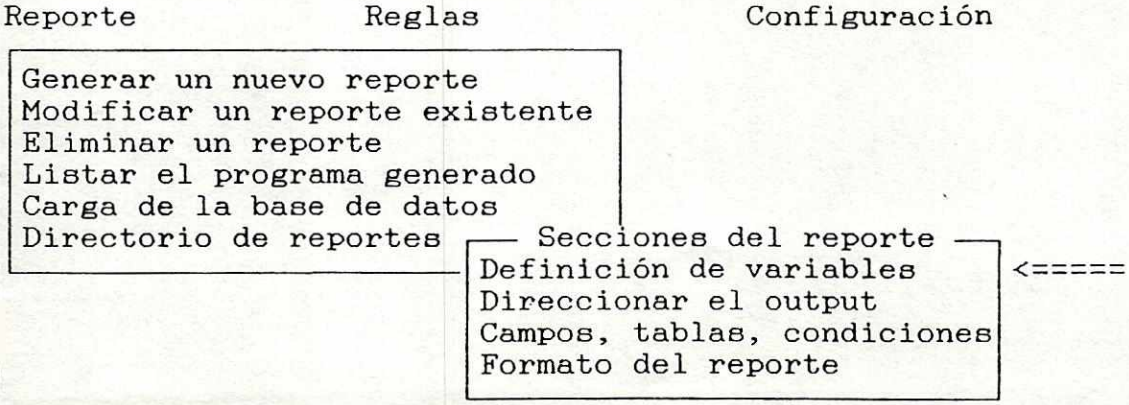
Pantalla No. 5

Esta ventana corresponde a la selección de la base de datos, en este ejemplo es Stores. Para seleccionarla, se coloca el cursor sobre la palabra, y se oprime la tecla de "enter". En seguida aparece un asterisco, que indica que fué seleccionada. Cerrar la ventana con la tecla "DEL".



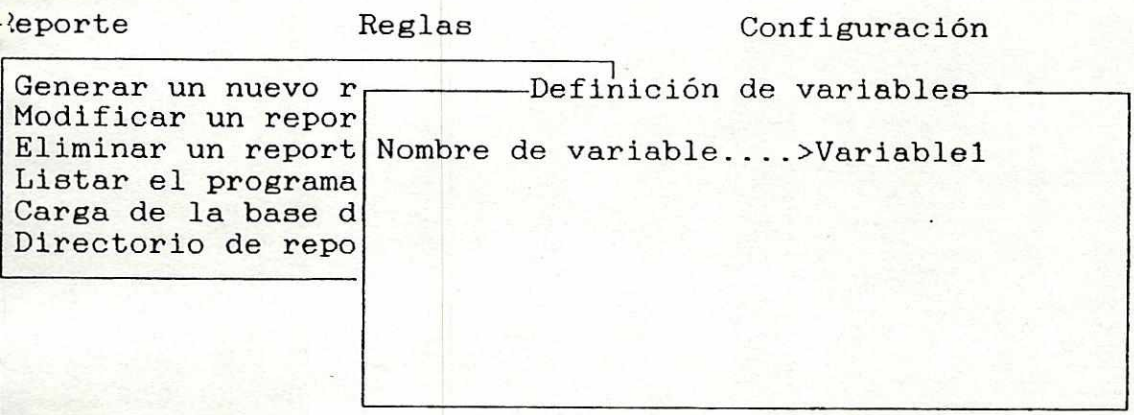
Pantalla No. 6

En seguida nos aparece la ventana que contiene las opciones de las secciones del reporte. Estas deben de elegirse de arriba hacia abajo.



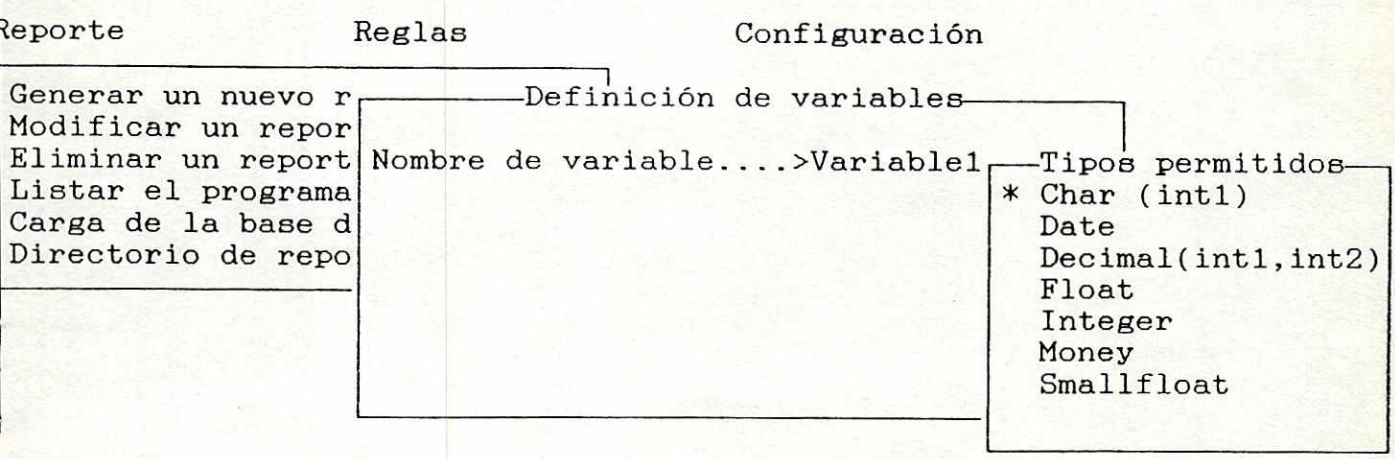
Pantalla No. 7

Al oprimir la tecla de "enter", sobre la definición de las variables, se activa la ventana donde se ingresa el nombre de la variable. Por omisión se inicializa con el nombre "Variable1".



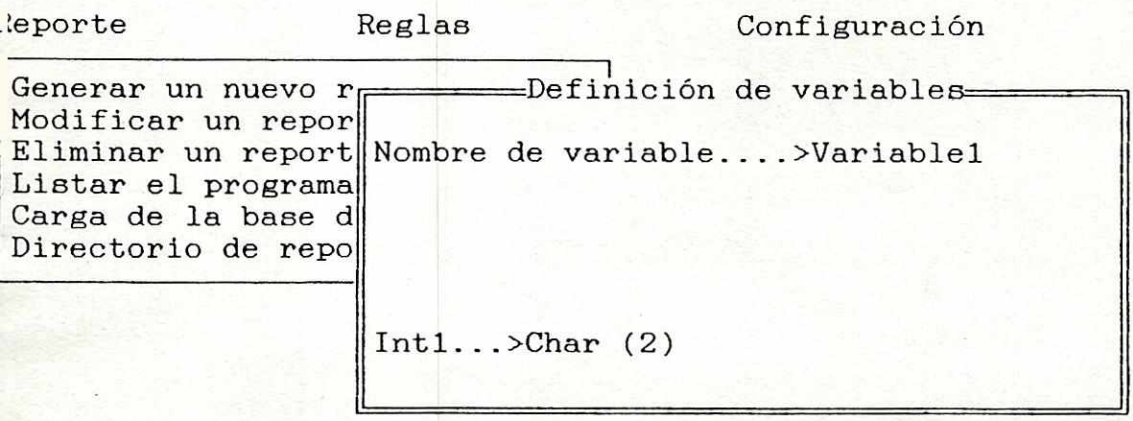
Pantalla No. 8

La variable1 debe de asignarsele el tipo correspondiente, mediante la ventana que se activa en seguida. Colocar el cursor sobre el tipo deseado y oprimir "enter".



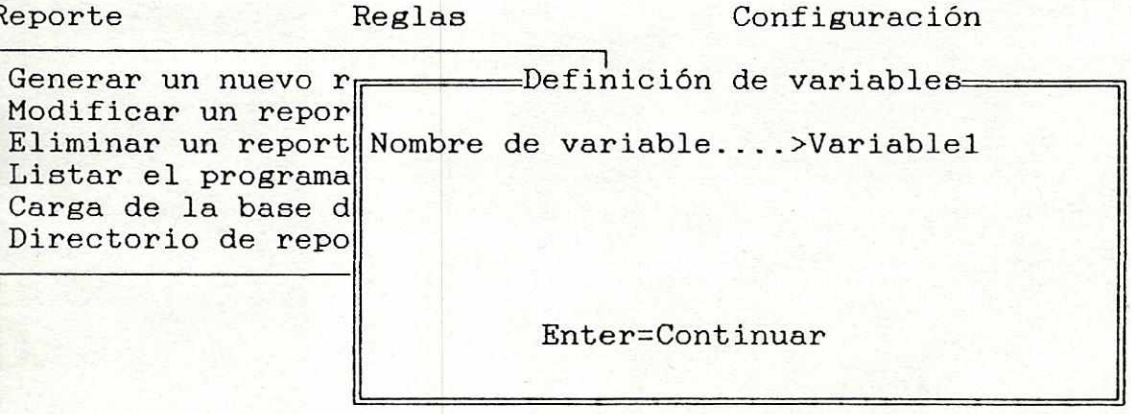
Pantalla No. 9

Debido a que se trata de un tipo de campo que requiere un ingreso de la dimensión, se pide la misma. En este caso es 2.



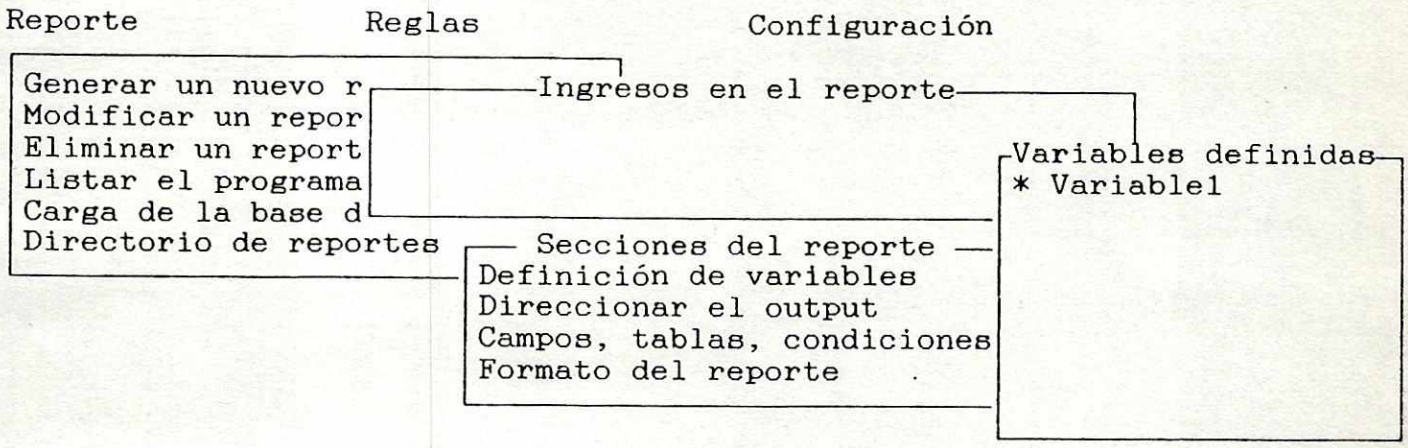
Pantalla No. 10

Ya definida completamente la variable, tenemos la oportunidad de seguir definiendo más variables, si oprimos la tecla de "enter" para el mensaje respectivo. Para terminar se desabilita la ventana oprimiendo la tecla "DEL".



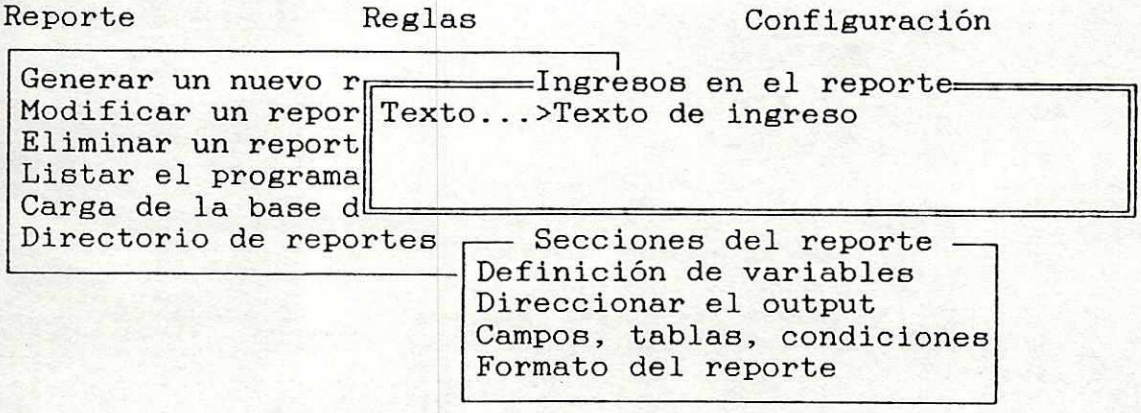
Pantalla No. 11

Dependiendo de las variables que hayan sido definidas, se activa una ventana donde el usuario selecciona a las que tendrán ingreso en el reporte.



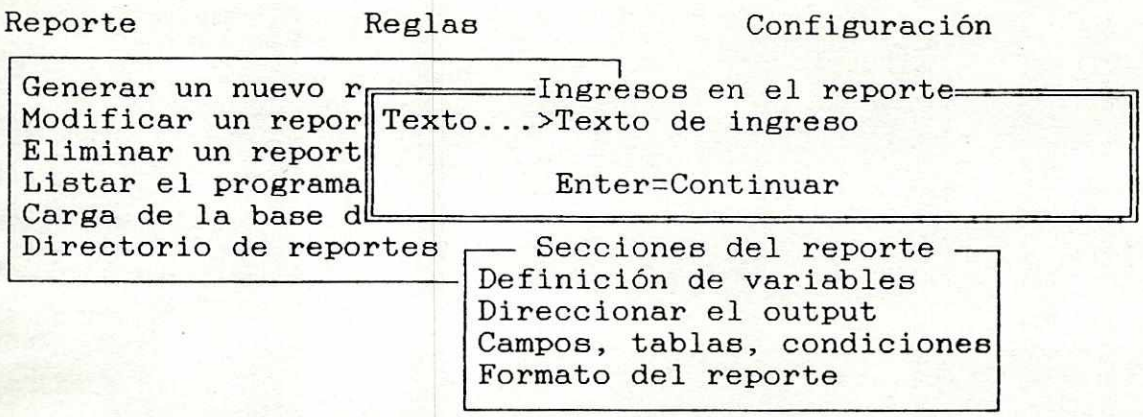
Pantalla No. 12

Se debe de ingresar el texto correspondiente para la variable. Por omisión se inicializa en "Texto de Ingreso", pero puede ser cambiado.



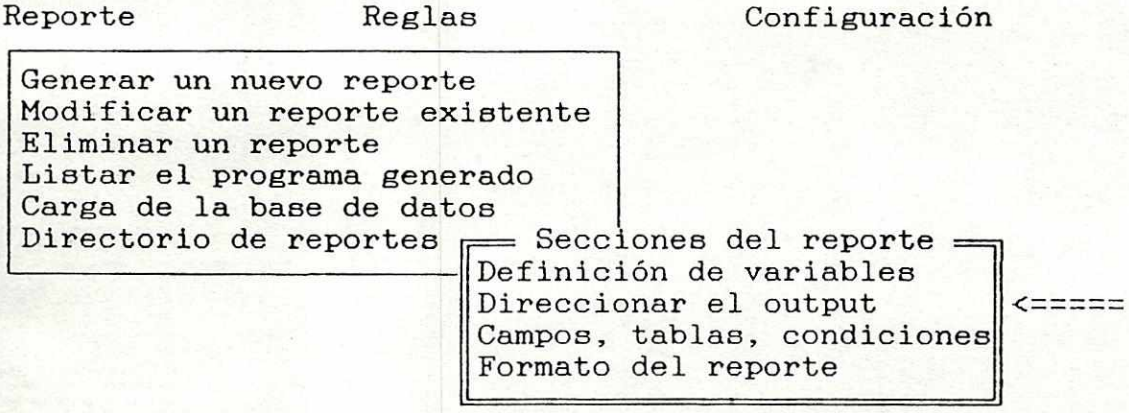
Pantalla No. 13

Para desactivar la ventana oprimir la tecla "DEL".



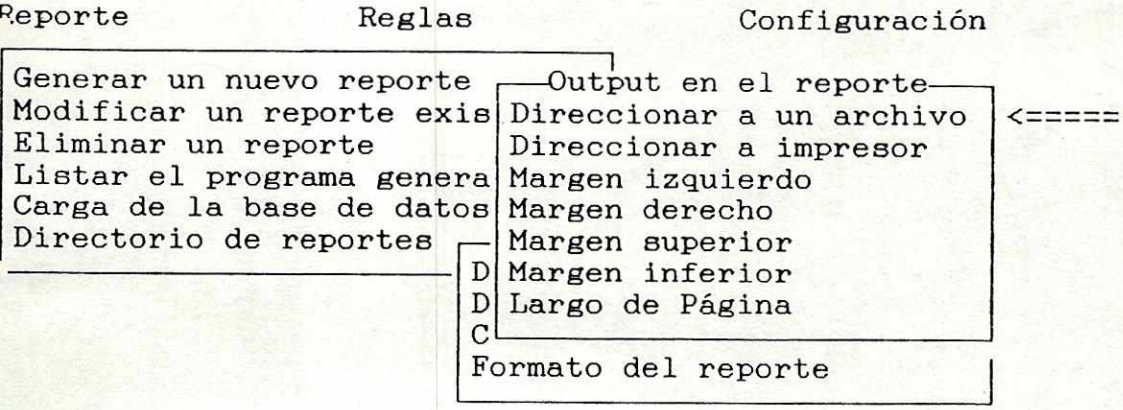
Pantalla No. 14

Ahora seleccionamos el direccionamiento del output, colocando el cursor como lo indica la flecha.

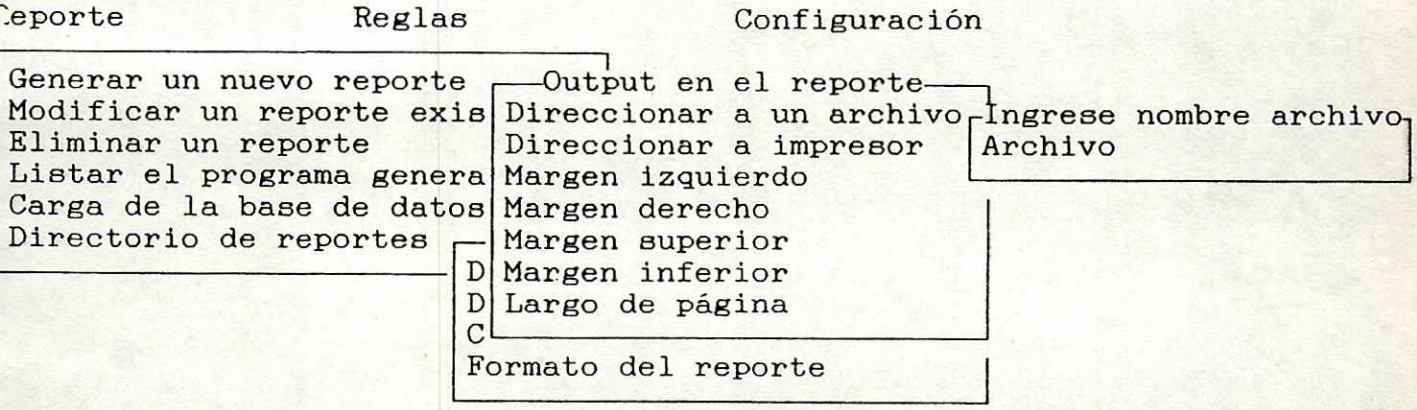


Pantalla No. 15

A continuación se activa la ventana que contiene las opciones del output del reporte. En este ejemplo, seleccionamos direccionar a un archivo y largo de la página. El usuario puede ingresar el nombre que desee. Por omisión es "Archivo".

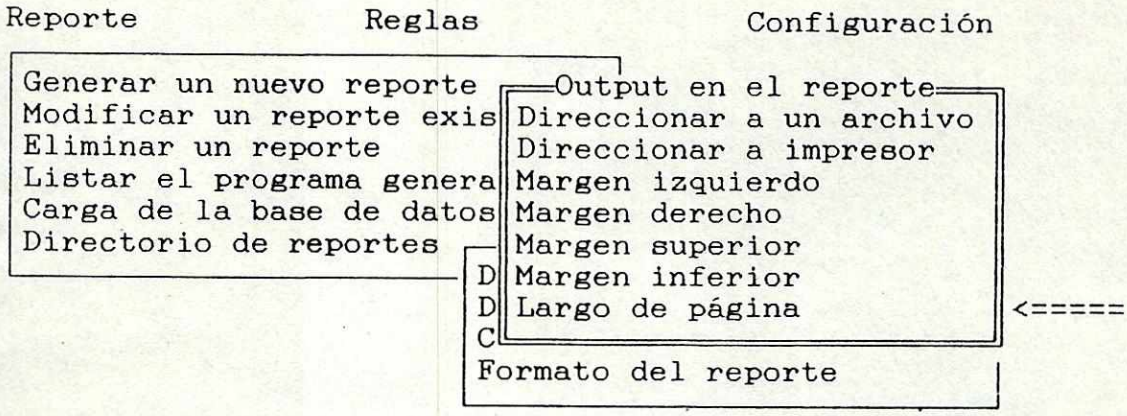


Pantalla No. 16

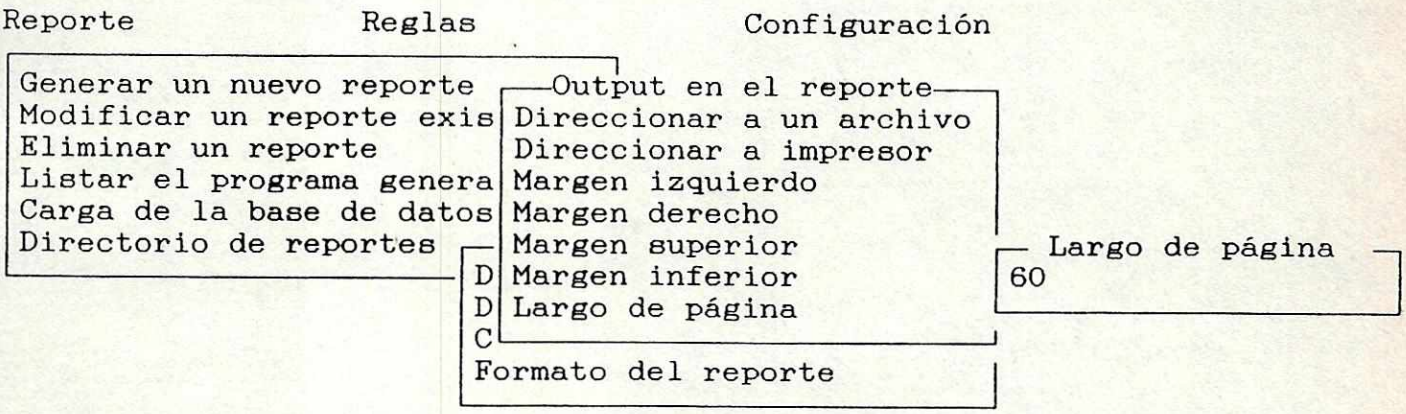


Pantalla No. 17

Para el largo de la página se inicializa en cero. En este ejemplo se ingresa 60.

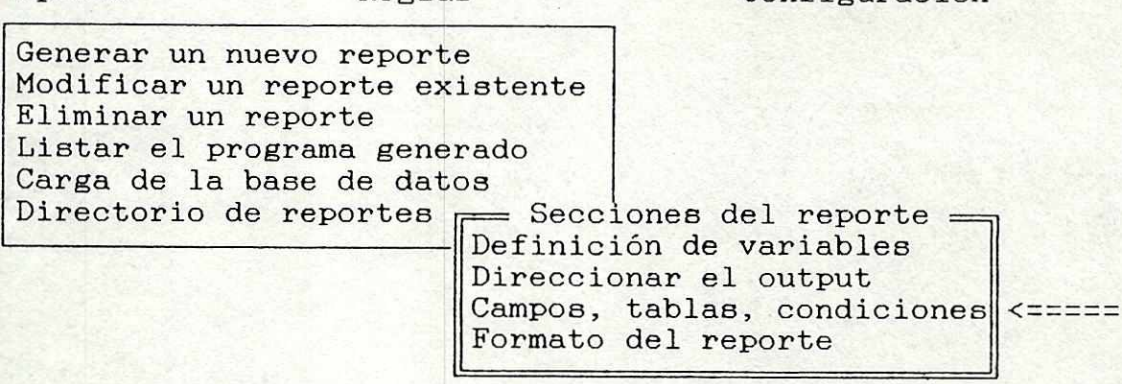


Pantalla No. 18



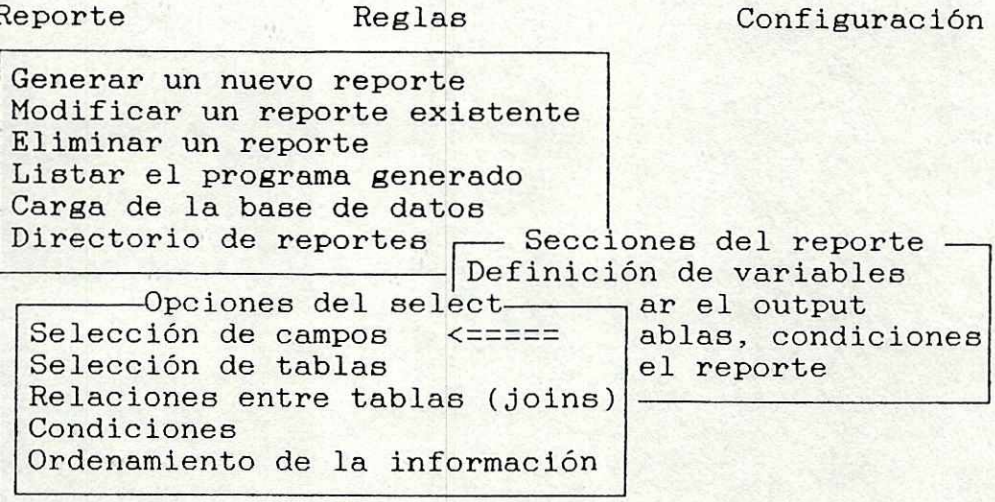
Pantalla No. 19

Después de desactivar la ventana del direccionamiento, regresamos a la ventana que contiene las opciones de las secciones. Ahora seleccionamos la opción de los campos, tablas y condiciones.



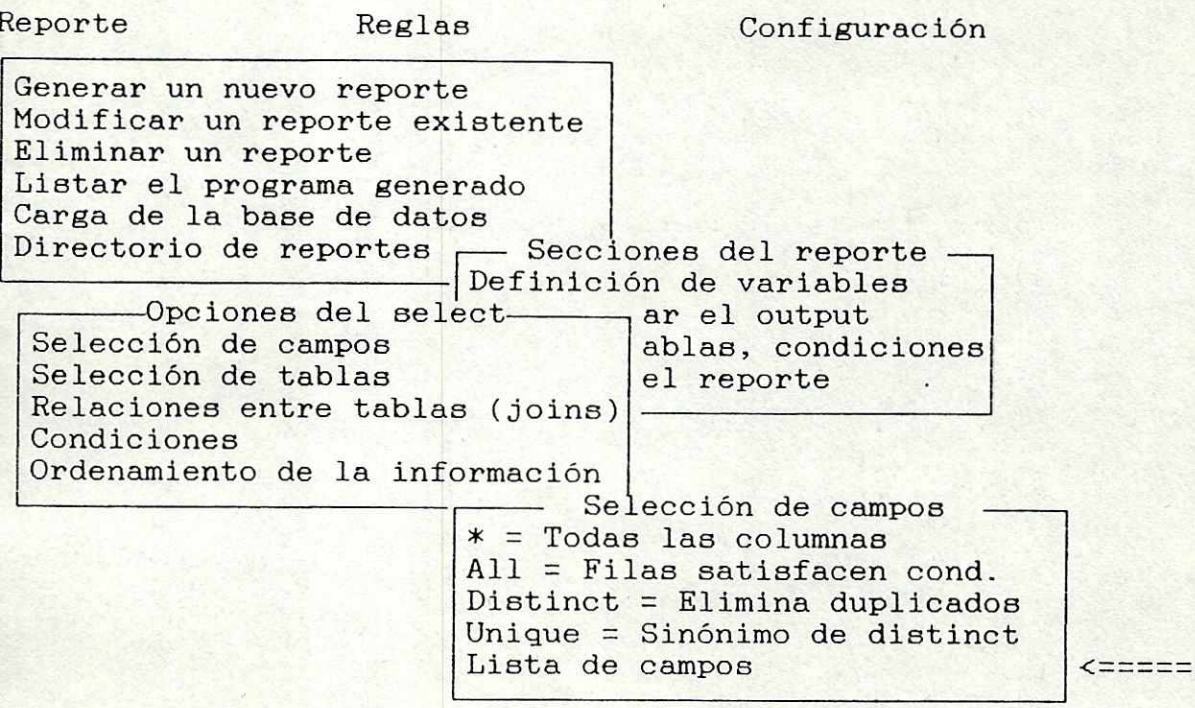
Pantalla No. 20

Las opciones de la sección SELECT aparecen en la ventana correspondiente. Seleccionamos tomando el orden de arriba hacia abajo, empezando por la selección de los campos.



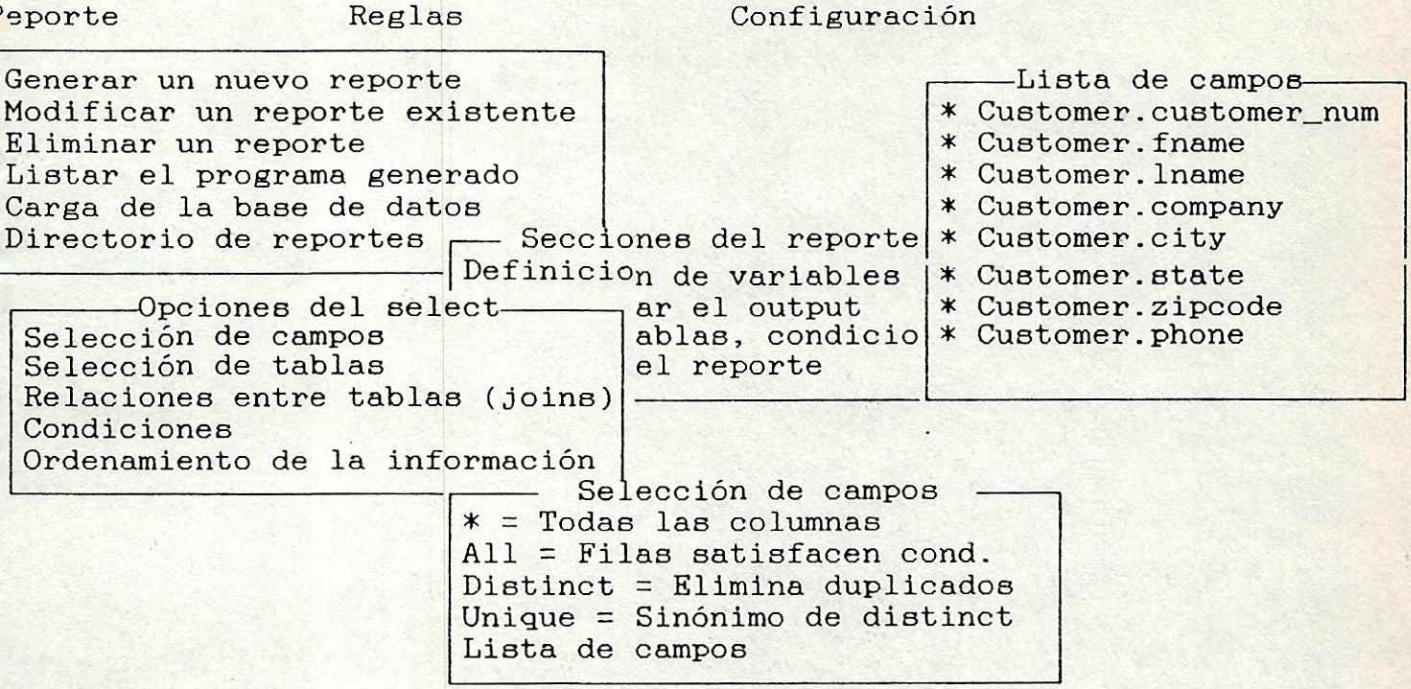
Pantalla No. 21

Las opciones para la selección de los campos se activa. En es caso se escoge la lista de campos.



Pantalla No. 22

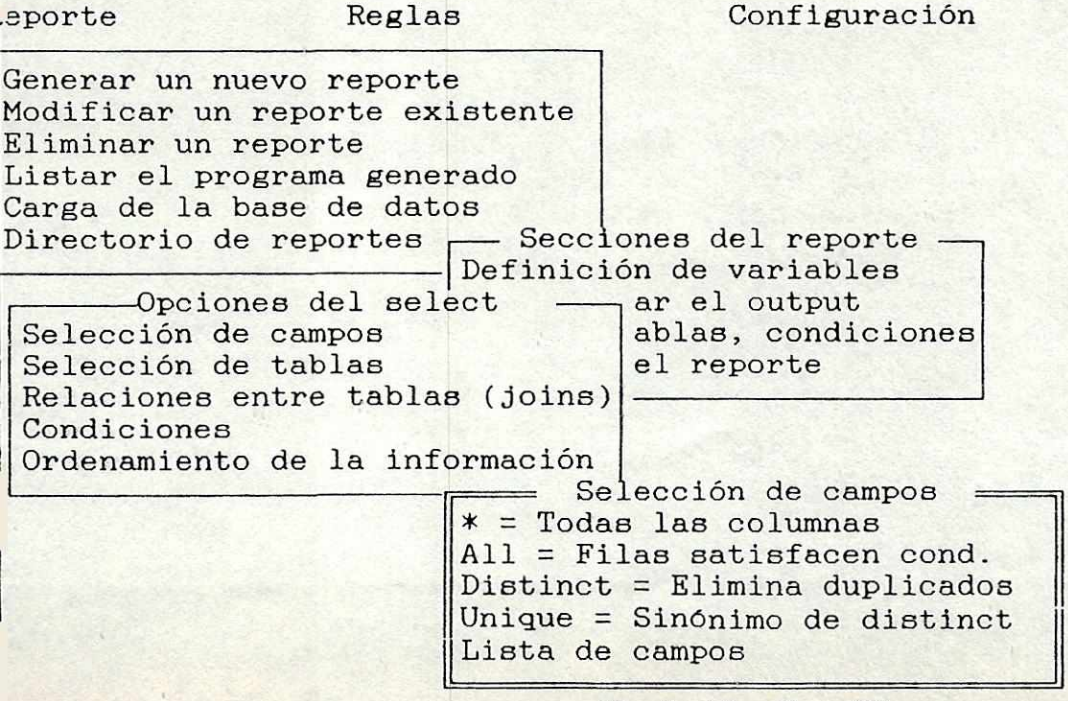
Al tener la ventana habilitada, se escogen los campos que se desean mediante la tecla de "enter", notando que al momento de escogerlos, aparece el respectivo asterisco.



Pantalla No. 23

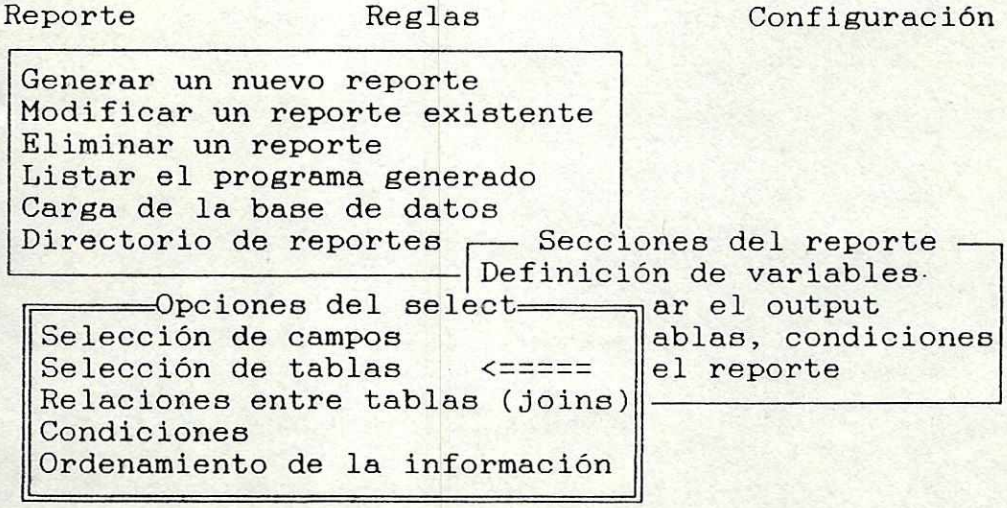
Al tenerlos todos marcados, cerramos la ventana con la tecla "DEL". Tan-

o la de la lista de los campos, como la selección de campos.

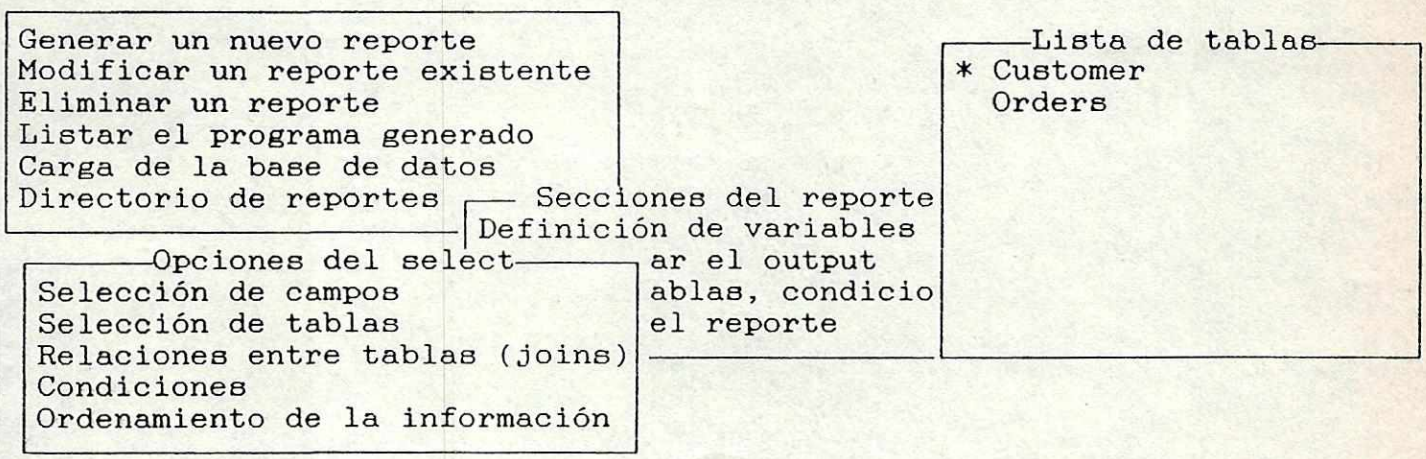


Pantalla No. 24

De nuevo en la ventana de la sección SELECT, escogemos las tablas que contendrá el reporte.

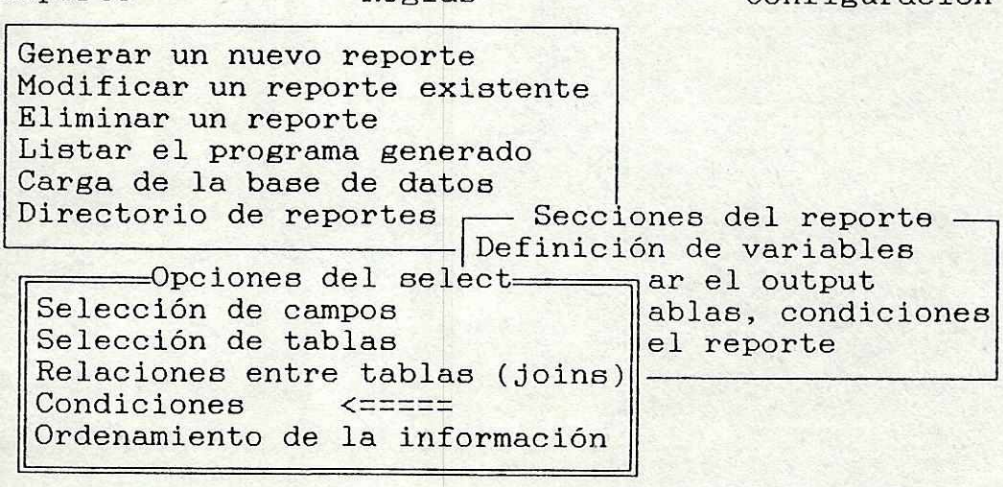


Pantalla No. 25



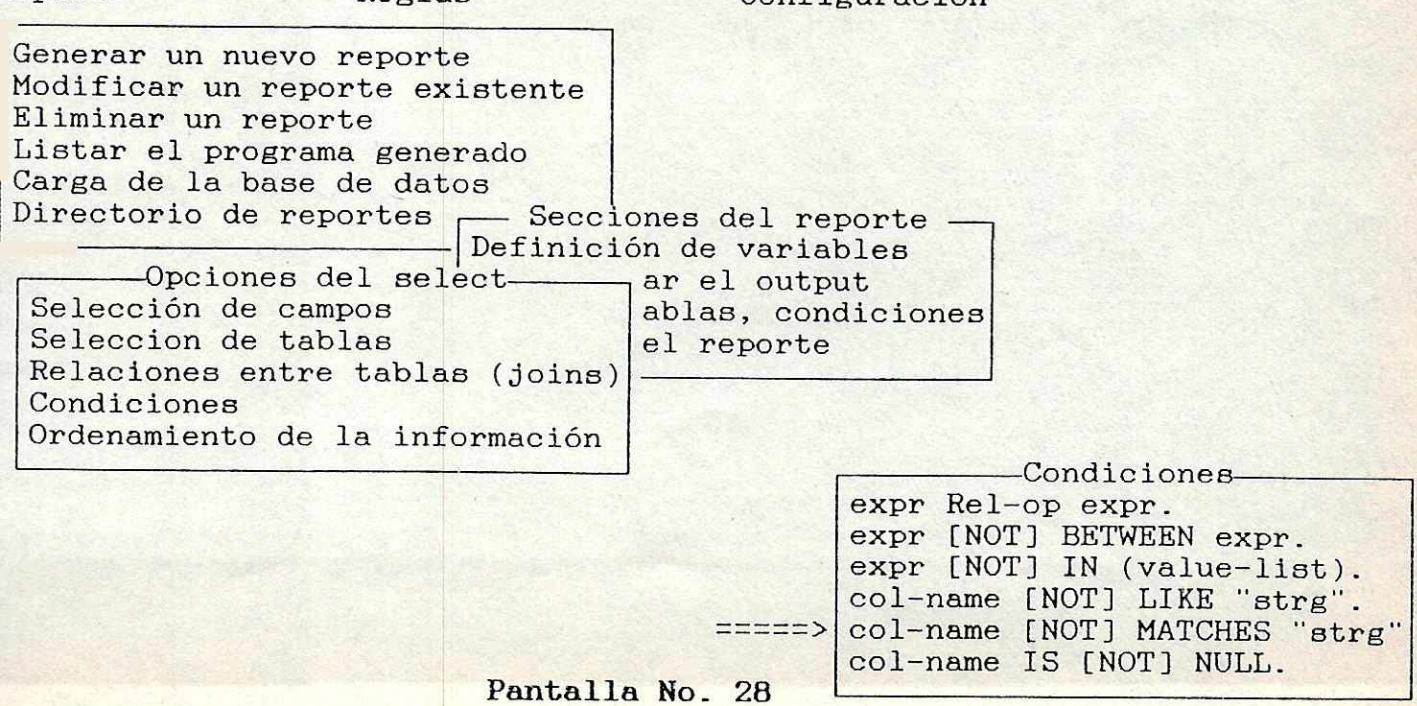
Pantalla No. 26

En este ejemplo, debido a que unicamente se escogió una tabla, la opción de relaciones entre tablas no tiene sentido, por lo que vamos a seleccionar la opción de las condiciones.



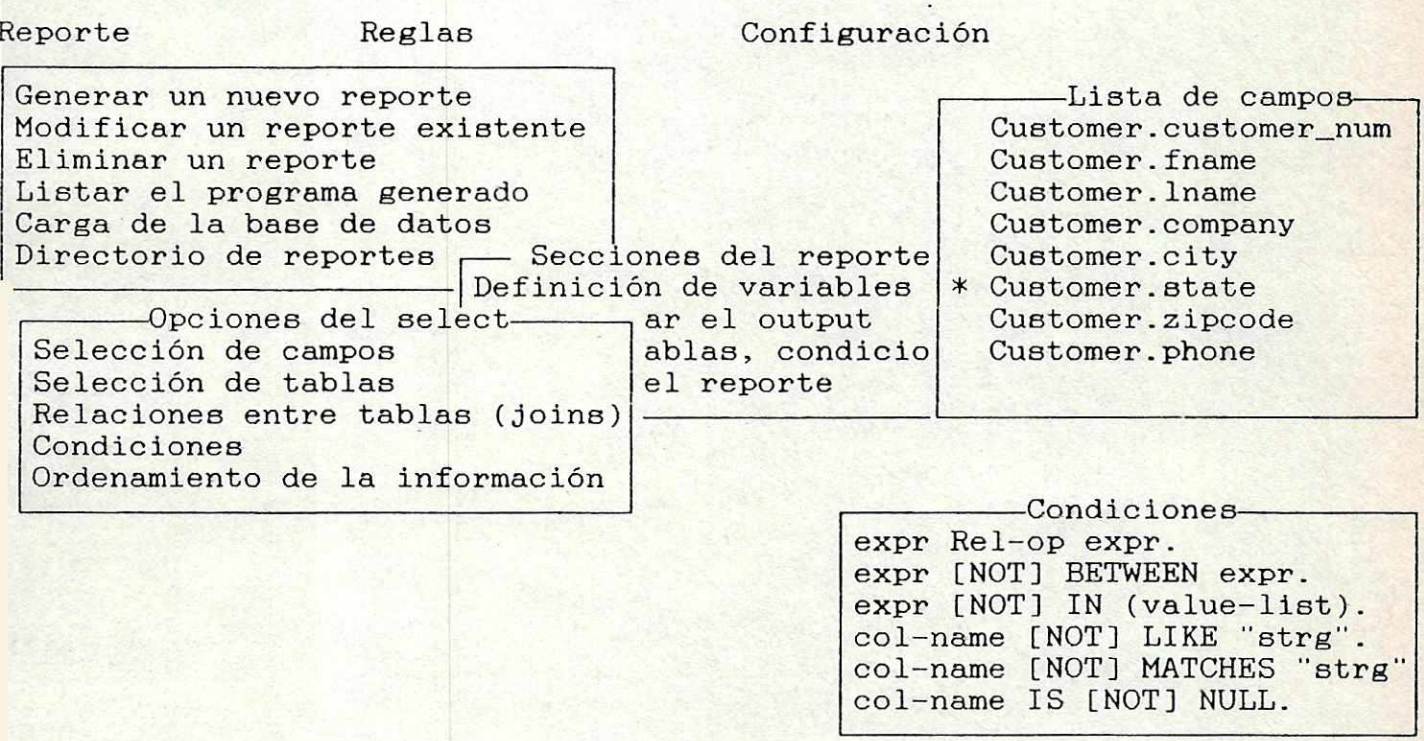
Pantalla No. 27

Al oprimir "enter" habilitamos la ventana que contiene las opciones respectivas de las condiciones.



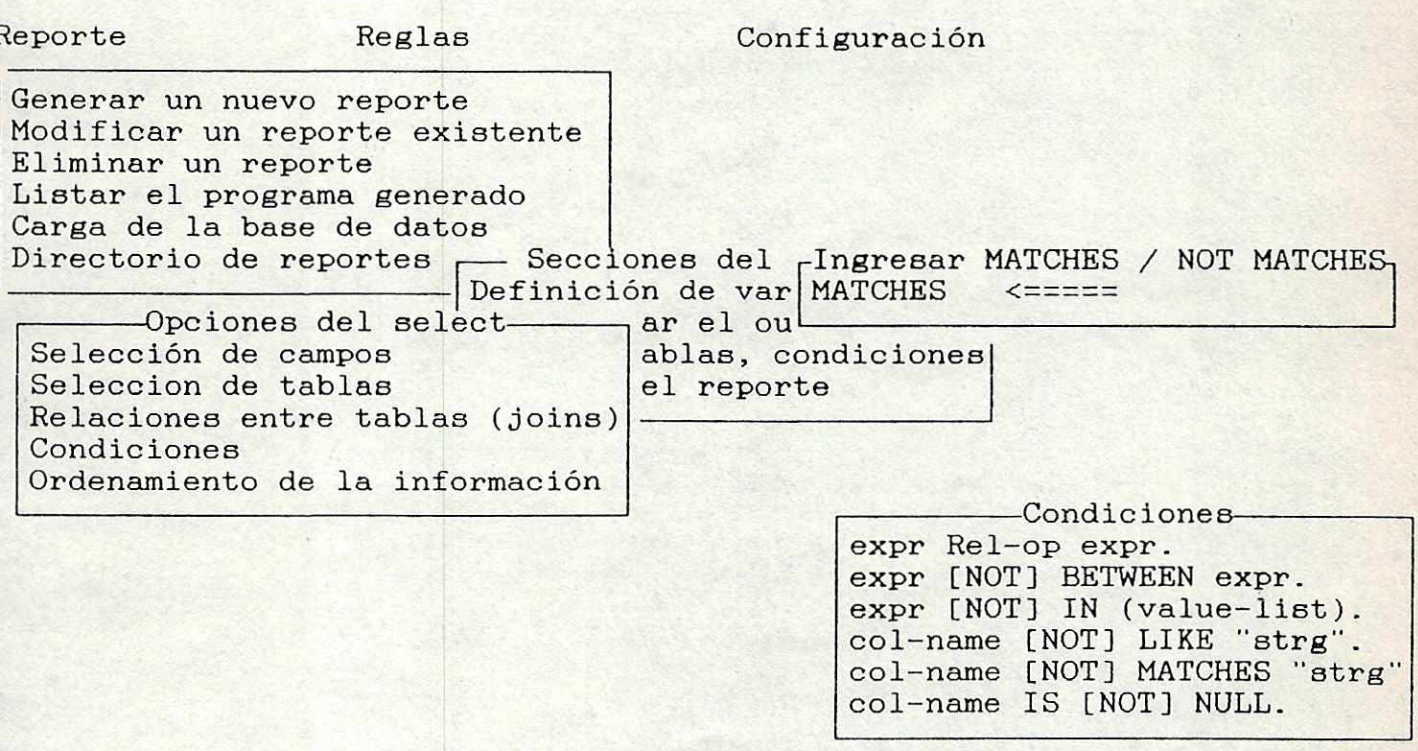
Pantalla No. 28

Debido a que la opción que se escogió requiere el nombre de campo de las tablas, se habilita una ventana donde se escoge por medio de las flechas del cursor, y oprimiendo "enter" en el campo deseado (Customer.state).



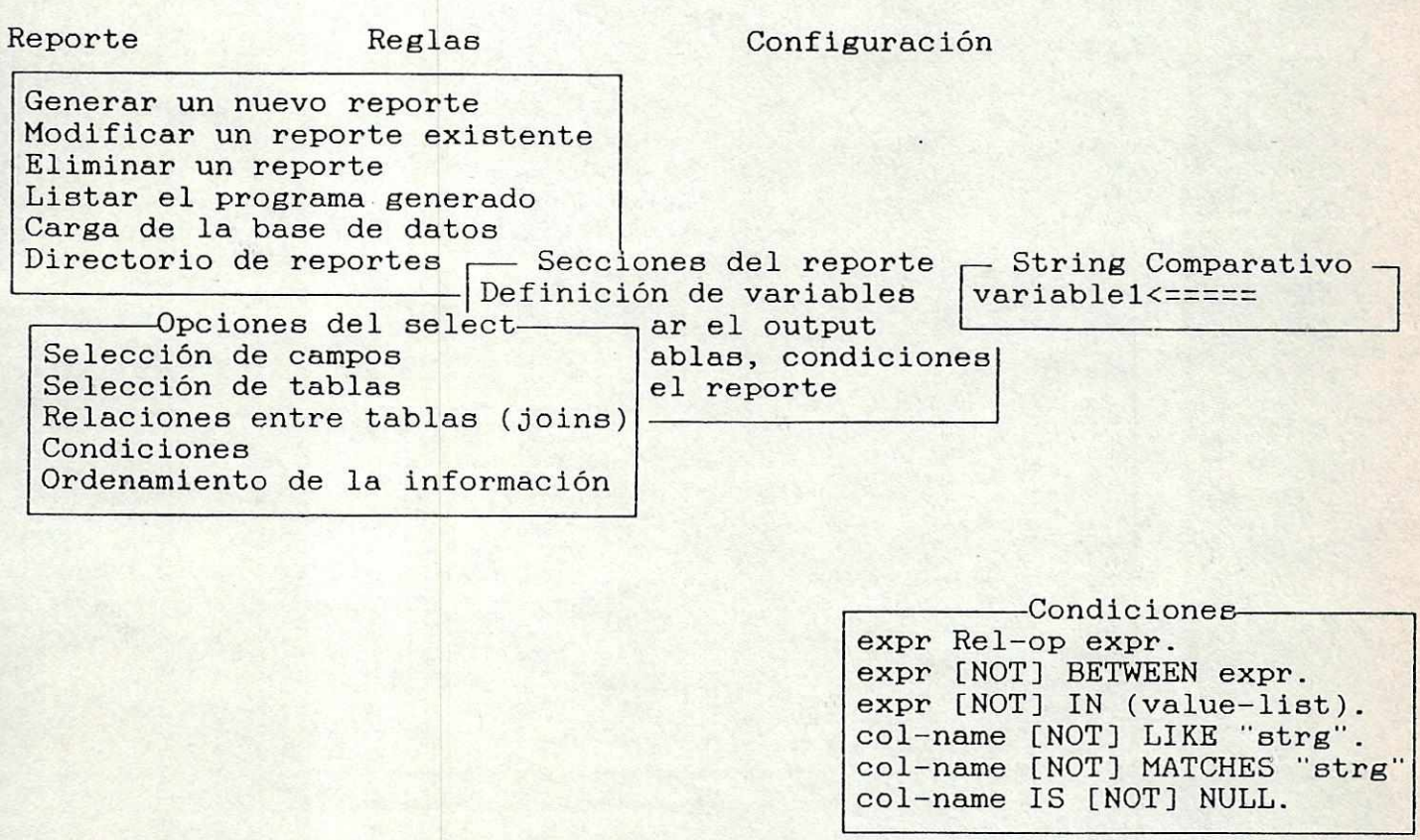
Pantalla No. 29

Ya que se tiene el campo seleccionado (asterisco a la par del mismo), se cierra la ventana con la tecla "DEL". En forma automática se habilita una ventana donde el usuario debe ingresar "MATCHES" o "NOT MATCHES" para que se inicie la formación de la condición que fué escogida. Oprimir la tecla de enter".



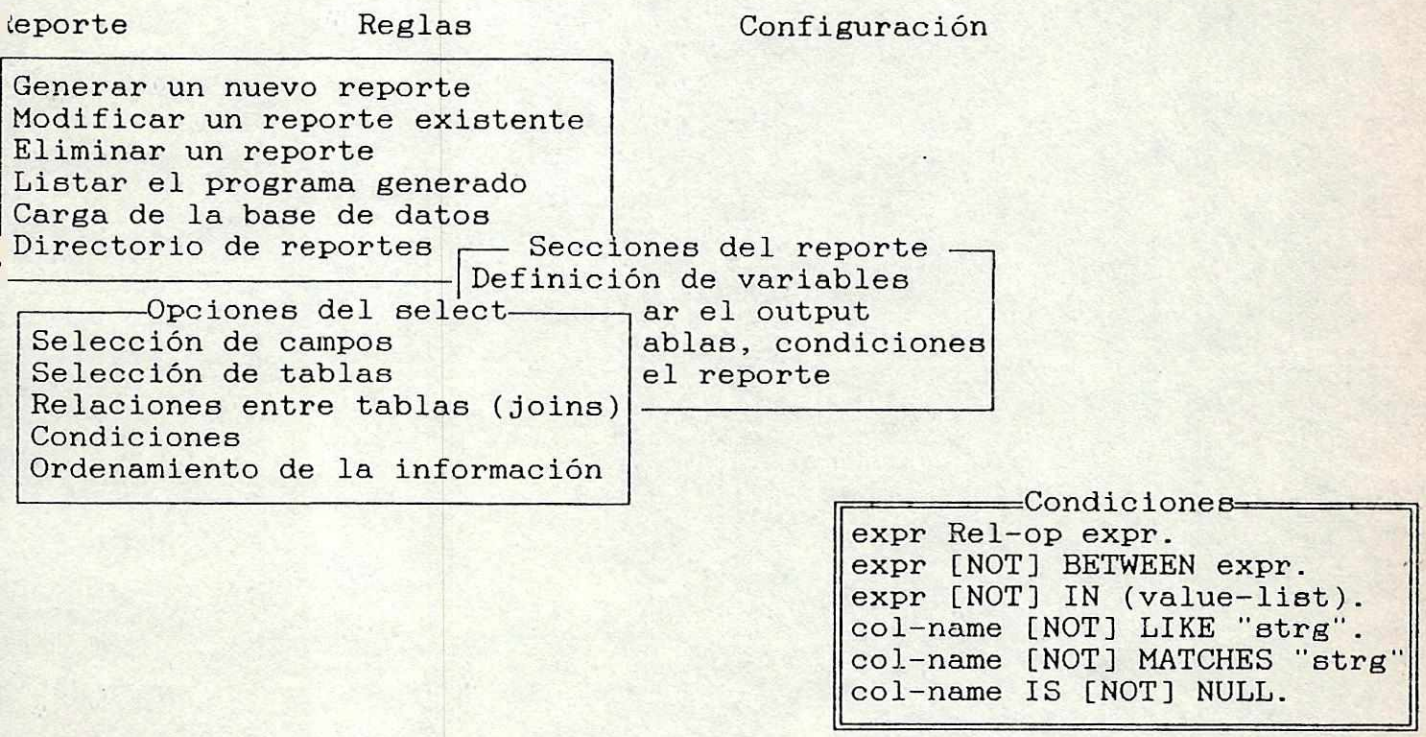
Pantalla No. 30

Finalmente se ingresa el string comparativo para terminar de completar la condición, y se oprime "enter". En este caso es variable1.

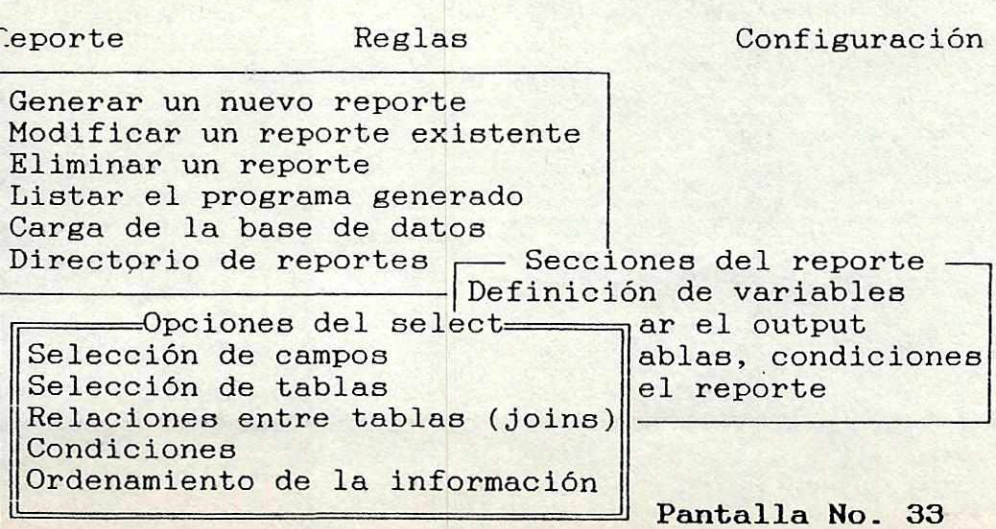


Pantalla No. 31

Ahora debemos cerrar las ventanas de las opciones de las "condiciones" y en seguida las opciones del "select". Ambas ventanas se desabilitan con la tecla "DEL".

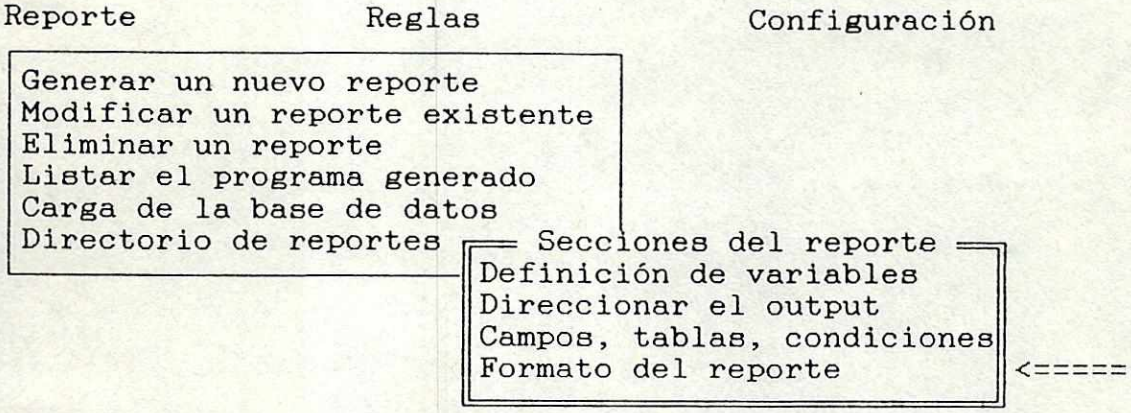


Pantalla No. 32



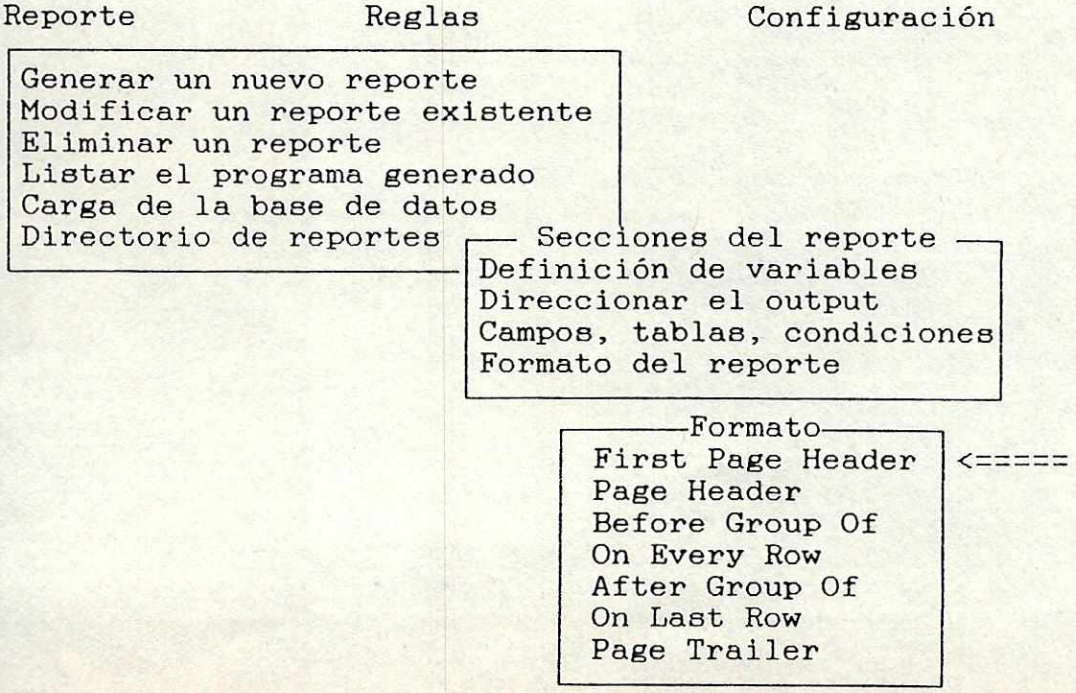
Pantalla No. 33

En el menú de las opciones de las secciones del reporte, seleccionamos el formato del reporte.



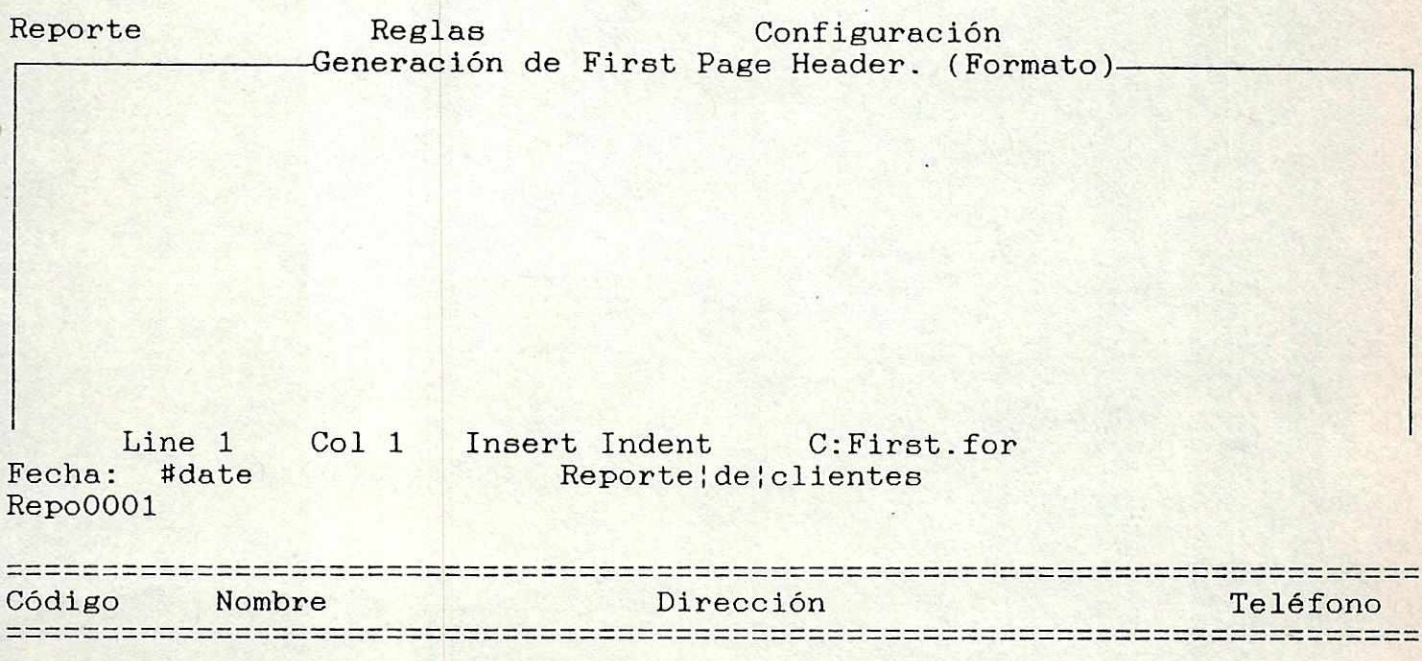
Pantalla No. 34

La ventana con las opciones del formato se habilita. Se escogen las opciones de arriba hacia abajo, empezando en este ejemplo por "First Page Header".



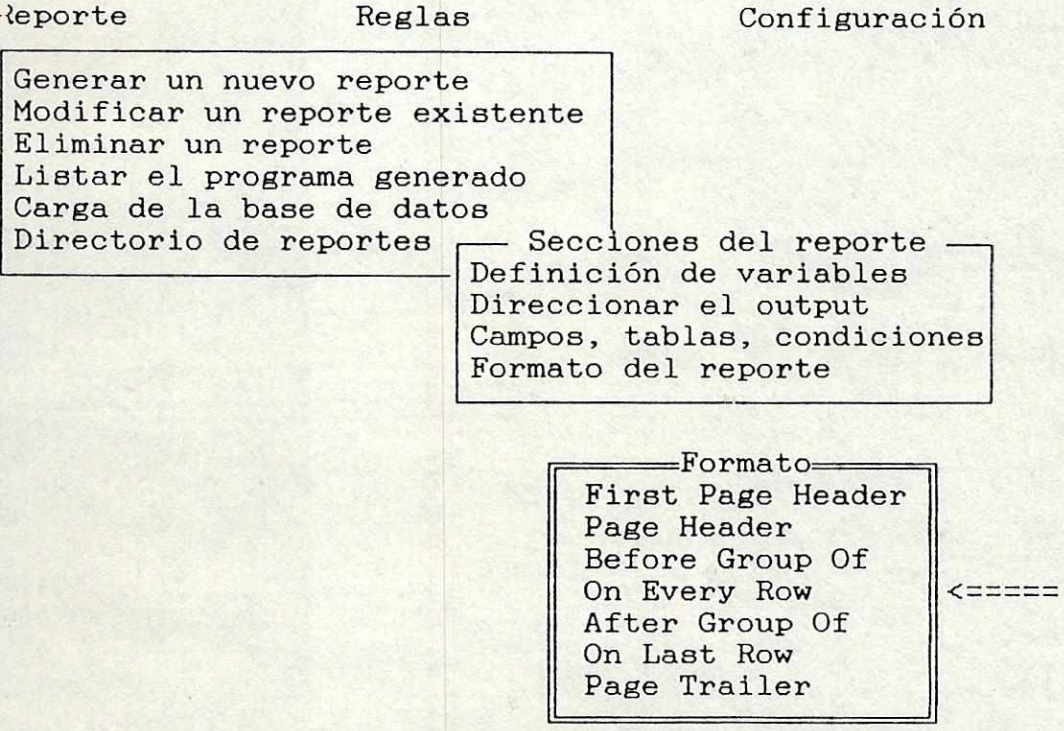
Pantalla No. 35

Las ventanas que se presentan corresponden a: la de la parte superior nos mostrará más adelante como es que llevamos el reporte y la de la parte inferior equivale al editor donde se dibujará el reporte y para terminar oprimiendo Ctrl Kd.



Pantalla No. 36

Después de salirnos del editor regresamos a las opciones del formato. Al seleccionar la opción que señala la flecha tendremos habilitadas las ventanas explicadas anteriormente, solo que ahora estaremos dibujando la sección "On Every Row".



Pantalla No. 37

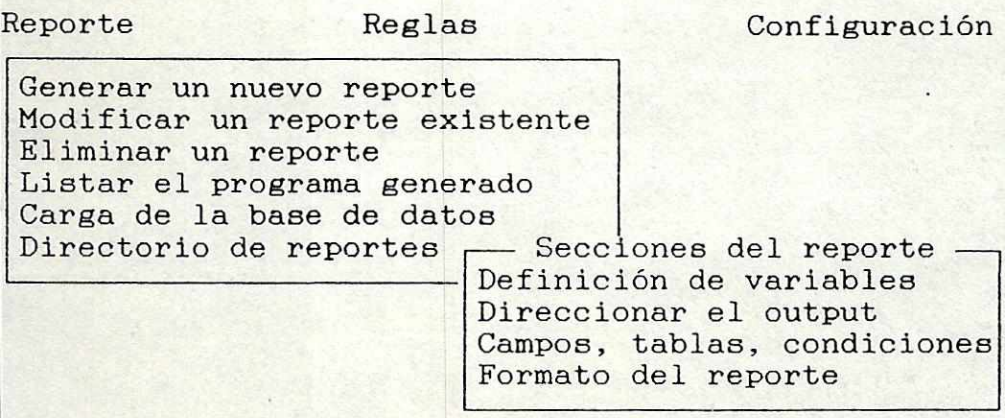
En el diseño de "On Every Row", los nombres de los campos utilizados se
orma de la inicial de la tabla a la que pertenece el campo y las cuatro
rimeras letras del nombre del campo. En nuestro ejemplo los nombres del
iseño equivalen a:

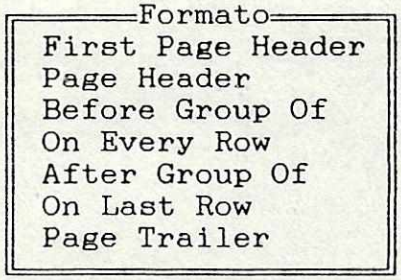
```
ccust.....> Customer.customer_num
 fnam.....> Customer.fname
ccomp.....> Customer.company
 city.....> Customer.city
 stat.....> Customer.state
 phon.....> Customer.phone
```

```
Reporte          Reglas          Configuración
  ----------------Generación de On Every Row. (Formato)----------------

 ===> FIRST PAGE HEADER <===
 Fecha:  #date                  Reporte¦de¦clientes
 Rep0001

 ================================================================
 Código     Nombre                  Dirección                Teléfono
 ================================================================


      Line 1    Col 1   Insert Indent     C:Every.for
#ccust    #cfnam , #clnam        #ccomp , #ccity , #cstat      #cphon
```

Pantalla No. 38

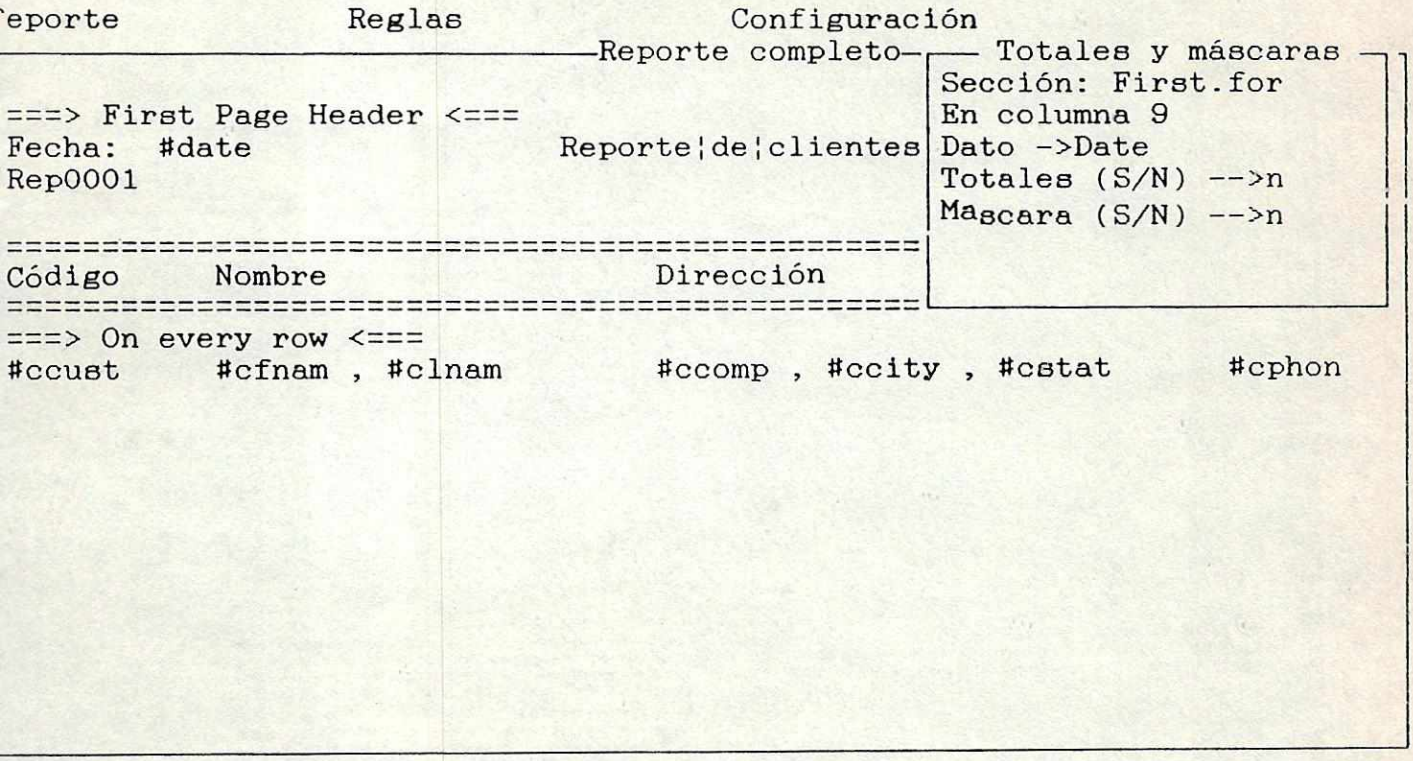
Al guardar el diseño, en seguida regresamos al menú de opciones del formato. En nuestro ejemplo, unicamente seleccionaremos las dos secciones anteriores, por lo cerraremos la ventana (Tecla "DEL").





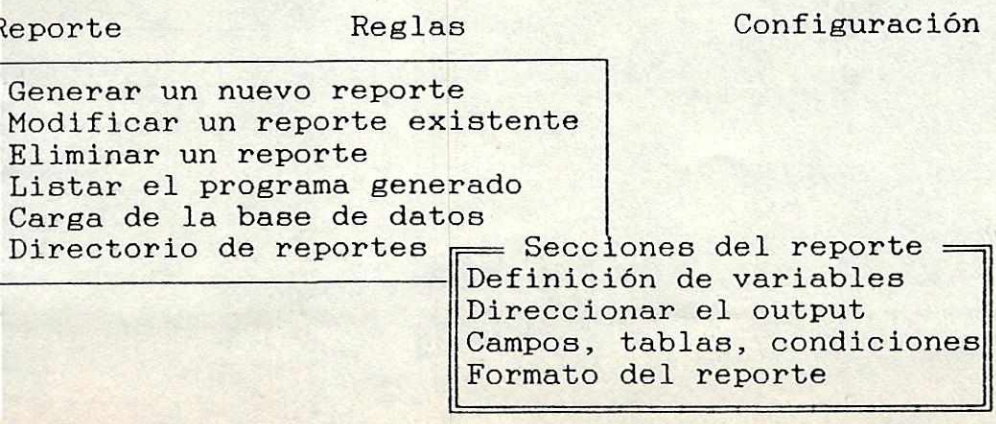
Pantalla No. 39

La siguiente ventana que se activa nos muestra como es que quedó el diseño total del reporte. Al mismo tiempo se activa una ventana en la esquina superior derecha que nos pide para cada campo del diseño, si deseamos que tenga totales o máscara. Esto lo va a hacer para todos los campos, es decir todos los que tengan antepuesto el signo "#".



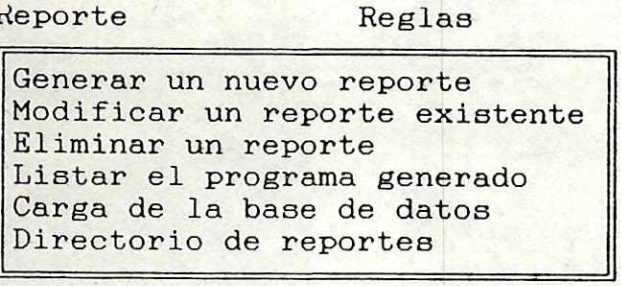
Pantalla No. 40

La ventana de los totales y máscara se pide a todos los campos. En la pantalla anterior unicamente se muestra un campo. Después que todos los campos hayan sido desplegados, regresamos al menú de secciones del reporte.



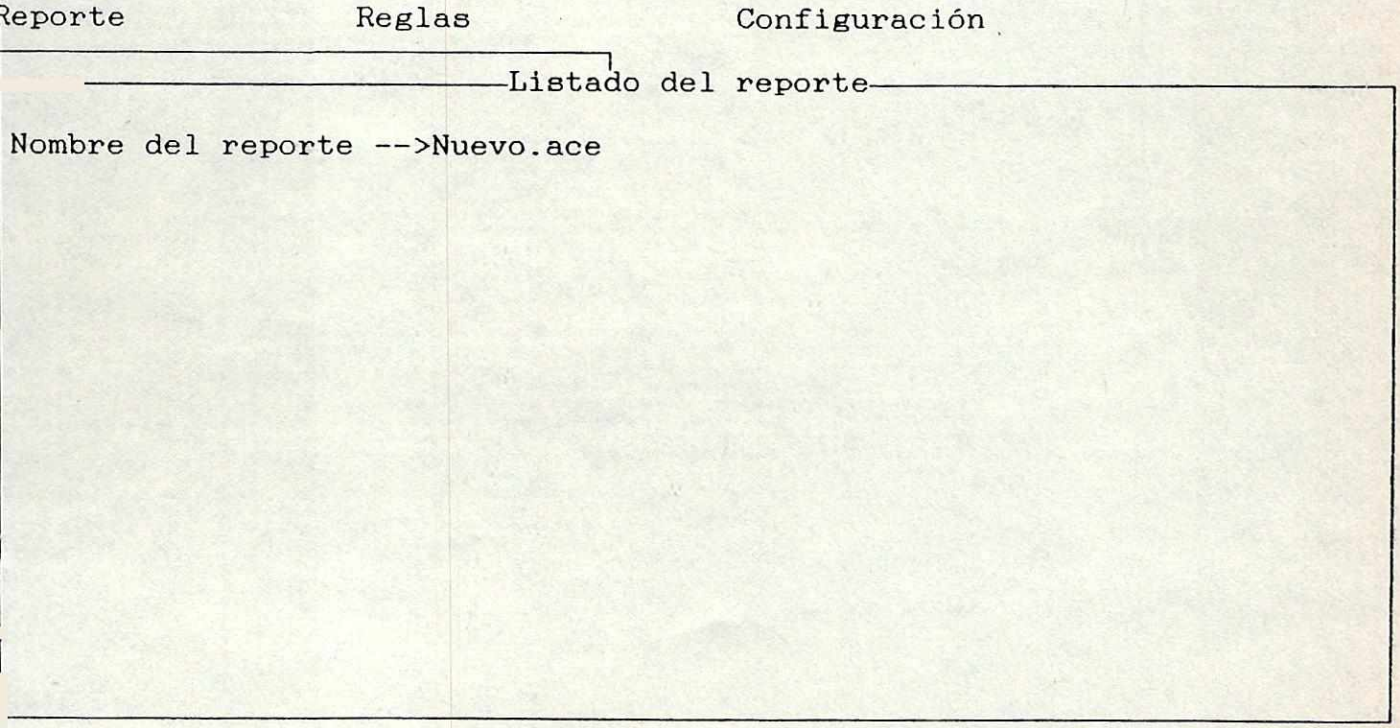
Pantalla No. 41

En este momento nuestro reporte ya fue generado completamente. Si deseamos ver como es que quedó, seleccionemos la opción "Lista el programa generado".



Pantalla No. 42

Ingresando el nombre del reporte, en nuestro ejemplo: nuevo.ace.



Pantalla No. 43

En las siguientes ventanas se presenta el resultado final del generador

Para continuar oprimir "enter".

Reporte Reglas Configuración

Listado del reporte

```
DATABASE
  STORES
END

DEFINE
  VARIABLE VARIABLE1 CHAR (2)
END

INPUT
  PROMPT FOR VARIABLE1 USING
  "Texto de Ingreso "
END

OUTPUT
  REPORT TO "ARCHIVO"
```

Pantalla No. 44

Reporte Reglas Configuración

Listado del reporte

```
  PAGE LENGTH 60
END

SELECT
CUSTOMER.CUSTOMER_NUM CCUST,CUSTOMER.FNAME CFNAM,CUSTOMER.LNAME CLNAM,CU
,CUSTOMER.CITY CCITY,CUSTOMER.STATE CSTAT,CUSTOMER.ZIPCODE CZIPC,CUSTOME
  FROM CUSTOMER
  WHERE CUSTOMER.STATE MATCHES $variable1
END

FORMAT

First Page Header
PRINT "Fecha: ", COLUMN 9, DATE , COLUMN 30, "REPORTE DE CLIENTES"
PRINT "REP0001"
```

Pantalla No. 45

```
Reporte              Reglas              Configuración
```

Listado del reporte

```
SKIP 1 LINES
PRINT "=============================================================
PRINT "CODIGO ", COLUMN 12, "NOMBRE ", COLUMN 35, "DIRECCION ", COLUMN 6
PRINT "=============================================================

On Every Row
PRINT CCUST  USING "####", COLUMN 12, CFNAM, COLUMN 19 ",", COLUMN 21 CL

END
```

<Enter>=Continuar

Pantalla No. 46

Al oprimir la tecla "enter" regresamos a las opciones del reporte. Para regresar al menú principal oprimimos la tecla "DEL".

```
Reporte              Reglas              Configuración
```

- Generar un nuevo reporte
- Modificar un reporte existente
- Eliminar un reporte
- Listar el programa generado
- Carga de la base de datos
- Directorio de reportes

Pantalla No. 47

Estando en el menú principal, por medio de la tecla "ESC", salimos del generador y se presenta el cursor del sistema operativo.

Reporte Reglas Configuración

<F1>-Ayuda -Cursor <Enter>-Opción <Del>-Cierra ventana <Esc>-Salir

Pantalla No. 48

C:\>

V. CONCLUSIONES

A. La utilización de sistemas basados en reglas, para la solución de problemas, es una técnica eficiente y flexible, y pienso que se puede emplear en la programación de proyectos de diferente naturaleza. Un sistema experto, si se diseña adecuadamente, proporcionará diferentes alternativas a seguir en determinado problema, basándose en el conocimiento de un experto en la materia.

B. Los generadores son herramientas que deben ser elaborados con más frecuencia, ya que ayudan a que el trabajo de los programadores o usuarios se facilite. La utilización del generador de reportes es bastante sencilla y la creación del reporte es rápida, por lo que se tiene un incremento de productividad al reducir el tiempo.

C. Los interfases basados en menús y ventanas permiten que la captura de información en la computadora sea comprensible para usuarios con poca experiencia.

VI. BIBLIOGRAFIA

Buchanan, B.; E. Shortliffe. Rule-based expert systems The mycyn experiments of the stanford heuristic programming project. USA, 1984 Addison Wesley Publising Company. 845 pp.

Hayes-Roth, F.; D. Waterman y D. Lenat. Building expert systems. USA, Addison-Wesley Publising Company. 1983 444 pp.

Jonhson, P. Expert systems. USA, Addison-Wesley 1985 Publising Company. 365 pp.

Koffman, E. Problem solving and structure programming in 1976 Pascal. USA, Prentice Hall. 430 pp.

Manual de referencia de Informix-SQL para DOS. USA, Re- 1986 lational Databases Systems, Inc. 390 pp.

Rice, J. Build program technique. USA, John Wiley & 1981 Sons Publication. 360 pp.

Rich, E. Artificial intelligence. USA, McGraw-Hill Book 1983 Company. 436 pp.

Waterman, D. A guide to expert systems. USA, Addison- 1986 Wesley Publising Company. 419 pp.

Winston, P. Artificial intelligence. 2a. ed. USA, Addi- 1984 son-Wesley Publising Company. 524 pp.

APENDICE A

Estructuras utilizadas en el generador

La estructura de datos en un programa es un aspecto vital, para que el desarrollo del mismo sea eficiente y es la forma como se guarda la información. En seguida se explicarán las estructuras utilizadas en los diferentes módulos del generador.

1. Listas encadenadas para la generación y modificación

En los módulos de generación y modificación se utiliza, principalmente, listas encadenadas. A continuación se explican los campos que tiene cada estructura:

Para la carga de la base de datos, se utilizó la siguiente estructura:

```
Tabla_Sch = Record
              Nombre_Sch : String[25];
              Tabla      : String[25];
              Tipo_Sch   : Char;
        End;
```

en donde:

- Nombre_Sch : puede tomar el valor de un campo, una tablas, la base de datos, o el tipo del campo.
- Tabla : se utiliza principalmente para los campos, para saber a cuál tabla pertenecen.

- Tipo_Sch : nos sirve para diferenciar el tipo de dato que almacena el registro, es decir si se trata de un campo, una tabla, el tipo del campo, etc.

Para tener opción a almacenar diferentes elementos de la carga de la base de datos, se definió la siguiente estructura:

```
Schema     : Array[1..Max_Schema] of Tabla_Sch;
```

Los datos leidos por el analizador de léxico, se almacenaban una lista doblemente encadenada. Se hizo doblemente encadenada, debido a que en ocaciones un cambio en un nodo repercutía en nodos anteriores, por lo que se tenía que contar con una estructura que pudiera ser leida para adelante y para atrás. Esta estructura se usa para la modificación de un reporte, y su definición es la siguiente:

```
Link =  ^Nodo;
Nodo = record
  Nombre_Actual : String[80];
  Nombre_Anterior : String[80];
  No_Token : Integer;
  Marca : Integer;
  Clases : Array [1..Max_Clase] of String[2];
  Next : Link;
  Prev : Link;
end;
```

en donde:

- Nombre Actual: guarda el valor que tiene el token actualmente. En un inicio es el valor leído por el analizador de léxico, más adelante cambia su valor, según lo que decida

el sistema experto. Este campo es el se escribe al nuevo archivo con cambios.

- Nombre Anterior: almacena el valor que tiene el token producto del analizador de léxico. No cambia su valor, básicamente se utiliza para buscar un token por su valor anterior.

- No_Token: almacena un número entero que identifica el token. La mayor parte del generador, hace refencias y comparaciones utilizando el No_Token.

- Marca: puede tener dos valores: verdadero (1) y falso (0). Sirve para saber si un nodo ya fué procesado por las reglas.

- Clases: sirve para identicar la sección a la que pertenece un token. Dentro de la sección a su vez puede tomar otros valores, como el caso de la sección FORMAT.

- Next y Prev: son apuntadores para otro nodo de la lista encadenada ya sea anterior o posterior al nodo actual.

Para generar el reporte, por cada sección se forma una lista encadenada, la cual se procesa en el sistema de reglas, para que produzca la correspondiente generación de la sección. Luego se elimina la lista, para que puede ser utilizada por la siguiente sección. La definición es como sigue:

```
Puntero =  ^Gen_Mod;
Gen_Mod = record
  Nombre_G_M : String[80];
  Seccion : String[2];
  Gen_Mod : Char;
  Topico : String[2];
  Prox : Puntero;
end;
```

en donde:

- Nombre_G_M: almacena el valor .de la variable o constante que el usuario selecciona o ingresa a través del interfase de menús y ventanas.

- Sección: identifica la sección a la que pertenece dicha variable o constante.

- Gen_Mod: identifica una generación y una modificación.

- Topico: dentro de la sección identifica diferentes aspectos como: si es una variable, un titulo, linea en blanco, una constante, etc.

- Prox: es el puntero al siguiente nodo de la lista encadenada.

2. **Reglas: archivo y procedimiento**

Para el interfase que existe entre Btrieve y turbo pascal, para que el archivo que almacena las reglas pueda tener mantenimiento, y para la generación de los procedimiento-regla, es necesario contar con las siguientes estructuras:

- **Registro del archivo de las Reglas:**

```
REGLAS_Record = Record
   Case Integer Of
   1: (Regla      : Packed Array[1.. 3] Of Char;
       Clase      : Packed Array[1.. 1] Of Char;
       Descrip    : Packed Array[1..50] Of Char;
       Condicion1: Packed Array[1..70] Of Char;
       Condicion2: Packed Array[1..70] Of Char;
       Condicion3: Packed Array[1..70] Of Char;
       Accion1    : Packed Array[1..75] Of Char;
       Accion2    : Packed Array[1..75] Of Char;
       Accion3    : Packed Array[1..75] Of Char;
       Accion4    : Packed Array[1..75] Of Char;
       Accion5    : Packed Array[1..75] Of Char;
       Accion6    : Packed Array[1..75] Of Char;
       Accion7    : Packed Array[1..75] Of Char;
       Accion8    : Packed Array[1..75] Of Char;
       Accion9    : Packed Array[1..75] Of Char;
       Accion10   : Packed Array[1..75] Of Char);
   2: (Start                 : Integer);
   End;
```

- **Llave del Archivo (Clase + Número de Regla):**

```
Key_Reglas_1 = Record
   Case Integer Of
   1: (Value          : Packed Array[1..4] Of Char);
   2: (Start          : Integer);
   End;
```

- **Nombre del Archivo:**

```
File_Name = Record
   Case Integer Of
   1: (Value          : Packed Array[1..25] Of Char);
   2: (Start          : Integer);
   End;
```

El campo "Start", en las diferentes difiniciones, es utilizado por Btrieve, para indicar el inicio de la variable dentro de los buffers manejados por él, y el campo "Value", es el valor de la variable misma.

Para la generación del procedimiento-regla, que se explicó en secciones anteriores, se utilizó la definición del registro-regla.

3. Arreglos y listas encadenadas para los menús y ventanas

El interfase de menús y ventanas cuenta con las siguientes definiciones:

- **Nodo que forma una Ventana:**

```
TitleStrPtr = ^TitleStr;          Puntero del Titulo.
WinRecPtr = ^WinRec;              Puntero de la Ventana.
WinRec = record
  Next: WinRecPtr;                Apuntador al próximo nodo.
  State: WinState;                Estado de la ventana.
  Title: TitleStrPtr;             Titulo de la ventana.
  TitleAttr, FrameAttr: Byte;     Atributos de pantalla para
                                  el título y el marco.
  Buffer: Pointer;                Ambiente completo de ventana.
end;
```

Para la utilización de este interfase de ventanas, fué necesario incluir la unidad de Ventanas (Unit Win), que trae turbo pascal.

- **Registro de un menú:**

```
Menu_Rec =  Record
              X,Y : Integer;          Filas y Columnas.
```

```
        Texto : String[30];     Texto u opción.
    End;
```

- **Arreglo de registro de menú: (para tener varias opciones)**

```
Menu = Array[1..10] of Menu_Rec;
```

- **Registro para seleccionar dentro de varias opciones:**

```
Sele_Rec =  Record
            Flag : Boolean;         Seleccionado o no.

            X,Y : Integer;          Filas y Columnas.

            Texto : String[30];     Opción.
        End;
```

- **Arreglo para selección de varias opciones:**

```
Selecc =Array[1..100] of Sele_rec;
```