CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS

DEL

INSTITUTO POLITÉCNICO NACIONAL

Rojo

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

SECCIÓN DE COMPUTACIÓN

SISTEMA PARA DISEÑO DE BASES DE DATOS EVE

Tesis que presenta el Ing. Raúl Hernández Stefanoni como requisito parcial para la obtención del grado de Maestro en Ciencias en la especialidad de Ingeniería Eléctrica. Trabajo dirigido por el Dr. Sergio V. Chapa Vergara

México, D.F. Marzo de 1994

CONTENIDO

1. INTRODUCCION

El sistema para diseño de bases de datos basado en el modelo Entidad-Vínculo-Extendido de Chen (SDBD-EVE) es una herramienta de software que facilita al usuario la creación de Base de Datos relacionales partiendo de la definición de un Modelo de Datos. Un modelo de datos no es lo mismo que una base de datos, ya que la base de datos física representa una interpretación del modelo de datos, el cual a su vez es una simplificación del Modelo de Empresa, el cual representa una descripción de alto nivel de la naturaleza de la organización y sus metas. De tal forma que una base datos física representa el último nivel de una serie de abstracciones del siguiente tipo : Mundo Real --> Modelo de Empresa --> Modelo de Datos --> Base de Datos. La creación del modelo de datos tiene la finalidad de formalizar en una estructura y reglas los datos de la empresa para que no sean vistos simplemente como campos dentro de archivos sino que proporcione información sobre el significado de los datos dentro de la organización para ello el modelado de datos incluye entidades que representan cosas de interés informacional para la organización, atributos que representan propiedades de dichas cosas y vínculos que expresan una relación explícita entre dichas cosas. En términos formales, un modelo de datos es una colección de conceptos matemáticos bien definidos que nos permiten estructurar los datos y expresar sus propiedades dinámicas y estáticas, y restricciones de integridad para una aplicación.

El SDBD -EVE siguiendo la filosofía anterior se sustenta en el modelo entidad-vínculo de Chen y sigue una metodología para el diseño lógico de la base de datos propuesta por Teorey, para garantizar que el modelo de datos que se va a implementar como un diseño físico de la base de datos sea no redundante y garantice la consistencia e integridad de la información al SDBD se le incorporó un modulo de normalización que es algoritmo de síntesis de relaciones propuesto por Bernstein [BERN76]. En el capitulo 2 se presenta el sustento teórico del SDBD incluyendo una descripción del modelo entidad-vínculo de Chen, una descripción de la metodología de Teorey para el diseño lógico de base de datos y el diseño general del SDBD. De los capítulos 3 al 7 se describen los componentes de SDBD : el diccionario de atributos, el diccionario y editor de iconos, la gramática de construcción , el interprete de traducción de diagramas de Chen a relaciones de Codd y los procesos de normalización y diseño físico. Por último en el capítulo 8 se presenta una descripción completa del funcionamiento del SDBD aplicado al diseño de una modelo de datos de ejemplo : SAPF (Sistema para Administración de Portafolios Financieros).

Por último cabe señalar que el SDBD-EVE forma parte de la línea de investigación del CINVESTAV denominada "Programación Visual" a cargo del Doctor Sergio V. Chapa Vergara y que el propósito de este trabajo es incorporar a LIDA (Lenguaje Iconográfico para el Desarrollo de Aplicaciones) [CHAP91] una herramienta que permita el diseño físico de base de datos relacionales.

2. DEFINICION DEL SISTEMA PARA DISEÑO DE BASES DE DATOS EVE

El Sistema para Diseño de Bases de Datos (SDBD) desarrollado tiene como objetivo crear una herramienta de software para el diseño físico de bases de datos relacionales. Se basa en la metodología propuesta por Teorey [TEOR86] para el diseño lógico de bases de datos, en el modelo ' Entidad-Vínculo' de Chen [CHEN76] y el algoritmo de normalización de Bernstein [BERN76].

En el campo de la ingeniería de software se tiene bastante experiencia en el desarrollo de modeladores de datos que permitan la integración de las cuestiones administrativas y técnicas. Dichos modeladores deben permitir entre otras cosas : delimitar los requerimientos de datos de la empresa, describir las acciones necesarias para modelar los requerimientos de datos y determinar el modelo conceptual de información que incluya los requerimientos planteados.

El objetivo final del modelador será la construcción de un 'Modelo de Empresa' donde queden expresados la planeación estratégica y los objetivos de la empresa, terminando con un modelo de base de datos que permita la implantación física de los requerimientos planteados. El modelo Entidad-Vínculo de Chen ha sido uno de los modeladores más utilizados en el diseño conceptual de base de datos y se ha convertido en una de las principales estrategias del mercado para el desarrollo de software [IBM89].

2.1. El modelo entidad-vínculo de Chen.

El modelo entidad-vínculo se ha caracterizado por ser una herramienta poderosa para la comunicación entre el diseñador y el usuario final durante el análisis de requerimientos y el diseño conceptual de bases de datos. La razón principal de su efectividad es su facultad de abstracción; usando entidades como abstracción de los atributos que conforman un objeto y enfocándose a los vínculos entre dichas entidades se reduce considerablemente el número de objetos en consideración y se simplifica el análisis.

El modelo entidad-vínculo, aunque modificado y extendido por otros, permanece como el principal modelo para el diseño conceptual de base datos. Inicialmente Chen propone tres clases de objetos : entidades, atributos y vínculos. Las entidades son el principal objeto para la representación de la información y usualmente denotan algún lugar, cosa o evento informacional de interés. Los atributos denotan las propiedades de las entidades definidas, como el color, el nombre, la edad o la fecha de contratación. Hay dos tipos de atributos: identificadores o llaves, que son utilizados para distinguir las ocurrencias de una entidad y los descriptores, que son utilizados para describir con detalle la ocurrencia de una entidad. Los vínculos representan las asociaciones de las entidades en el mundo real, tienen un significado semántico e indican la conectividad entre las ocurrencias de entidades ('1 a 1', '1 a muchos' y 'muchos a muchos').

Las extensiones que se han realizado consisten en la introducción de significados semánticos y categorías dentro del modelo entidad-vínculo, resultando en dos tipos adicionales de objetos: jerarquía de generalización y jerarquía de especialización. La jerarquía de especialización especifica la posibilidad de sobreposición de subconjuntos, mientras la jerarquía de generalización no permite la sobreposición.

2.2. Metodología de Teorey para el diseño lógico de bases de datos.

La metodología propuesta por Teorey simplifica el diseño de bases de datos relacionales grandes, reduciendo el número de dependencias a ser analizadas, a través de la incorporación de un paso de diseño conceptual de la base de datos con el uso del modelo entidad-vínculo de Chen. El diagrama entidad-vínculo generado se transforma al modelo relacional de Codd preservando la integridad de los datos a través de normalización. Los pasos básicos de la metodología son :

- **Paso 1: Análisis de requerimientos con el modelo Entidad-Vínculo-Extendido (EVE).** El análisis de requerimientos se realiza en estrecha comunicación con el usuario final, y consta básicamente de :
  - √ Identificación y clasificaciones de entidades y atributos.
  - √ Identificación de jerarquías de generalización y especialización.
  - √ Definición de vínculos entre las entidades.
  - √ Integración de diferentes vistas de entidades, atributos y vínculos.
  - √ Creación del modelo de empresa y el diseño conceptual de la base de datos.

- **Paso 2: Transformación del diagrama EVE al modelo relacional de Codd.** Sobre la base de una categorización de los constructores del modelo entidad-vínculo-extendido y un conjunto de reglas de mapeo, cada vínculo y sus entidades asociadas, son transformados en un conjunto de relaciones de Codd eliminando las relaciones redundantes.

- **Paso 3: Normalización de relaciones.** Para cada relación derivada del diagrama EVE se analizan sus dependencias funcionales y se normaliza al más alto grado posible usando cualquier técnica de normalización. Nosotros emplearemos el algoritmo de Bernstein para normalización a tercera forma normal.

- **Paso 4: Diseño físico de la base de datos.** El diseño físico es muy dependiente del manejador de bases de datos donde se implemente el diseño, para nuestro caso se usará el manejador de base de datos definido para LIDA [CHAP91].

2.3. Diseño general del SDBD

El sistema para diseño de bases de datos (SDBD) que presentamos está constituido por una n-tuple de elementos como se muestra a continuación :

SDBD (DA, DEI, GC, INT, PRO)

Donde :

DA : **Diccionario de Atributos.**
DEI: **Diccionario y Editor de Iconos EVE**
GC: **Gramática de Construcción.**
INT: **Intérprete.**
PRO: **Procesos.**

El detalle de cada uno de los componentes del SDBD se presenta a continuación.

3. DICCIONARIO DE ATRIBUTOS (DA)

El diccionario de atributos es una tabla donde se centraliza la información que caracteriza todas las propiedades de las entidades que forman el diagrama entidad-vínculo. Cada atributo refleja una propiedad especifica de una entidad y se representa de la siguiente manera :

Nombre-atributo, Tipo-atributo, Tamaño-atributo, Edición

Donde Nombre-atributo es un identificador del atributo. Tipo-atributo representa el tipo de información del atributo y puede ser :

- √ A: Alfanumérico.
- √ D: Fecha.
- √ C: Numérico de punto flotante
- √ N: Numérico entero,(blancos a la izquierda)
- √ Z: Numérico entero (ceros a la izquierda)

Tamaño-atributo representa la longitud del atributo y Edición representa la máscara de captura para dicho atributo y el caracter "_" representa un posible espacio de edición.

El diccionario es delimitado por las palabras reservadas #diccionario y #fin diccionario, de tal forma que un diccionario de atributos común sería :

```
#diccionario
        Nombre, A,      15,     "_______________"
        Edad,   N,      2,      "__"
        Sueldo, C,      12,     "$_________.__"
#fin diccionario
```

El diccionario de atributos se procesa por separado y se analiza a través de la siguiente gramática de reconocimiento

Gramática del diccionario de atributos

<Diccionario> ::= #dicionario <atributos> #fin diccionario
<atributos> ::= <sentencia><sentencias>
<sentencia> ::= <identificador> , <tipo> , <longitud>, <máscara>
<sentencias> ::= <sentencia> <sentencias> | ε
<longitud> ::= <número> <longitudes>

<longitudes> ::= <número> <longitudes> | ε
<máscara> ::= " <caracteres> "
<identificador> ::= <letra> <caracteres>
<caracteres> ::= <caracter> <caracteres> | ε
<tipo> ::= A | D | N | Z | C
<letra> ::= a | b | c | ... | z | A | B | C | ... | Z
<número> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<caracter> ::= <letra> | <número> | _ | . | / | (|) | * | $ | % | & | @

Cabe señalar que el diccionario de atributos definido aquí para el SDBD tiene la propiedad de ser un **diccionario central** lo cual significa que todos los atributos de cualesquiera base de datos que se defina tendrán una especificación única. Adicionalmente la gramática del diccionario permite adicionar propiedades a cada atributo, como validaciones, características de impresión, definición de captura obligatoria, etc, propiedades que pueden ser incorporadas a muchos manejadores de base de datos que utilicen diccionarios centralizados.

3.1. Anotaciones sobre la implementación de Diccionario de Atributos en lenguaje C

El diccionario de atributos está formado por un editor de texto, una interfase con el usuario a través de ventanas y un compilador que se basa en la gramática propuesta para el diccionario. En esta parte describimos exclusivamente la parte que corresponde al compilador porque es la que mayormente nos interesa :

El diccionario de atributos es controlado por una arreglo de elementos cuyo tamaño máximo=MXELE y cada elemento está defino por la siguiente estructura de datos :

```
struct dict {                          /* Diccionario de datos   */
   char dename [NAMLEN+1];             /* nombre                 */
   char detype;                        /* tipo                     */
   int delen;                          /* longitud                 */
   char *demask;                       /* mascara de captura       */
} dc [MXELE];
```

Este módulo cuenta con unas cuantas funciones generales que sirven para controlar la entrada y salida del programa y los errores en la gramática de construcción, a continuación definiremos de forma general las funciones de mayor interés :

static void error(int n)

Recibe como entrada el número de error ocurrido y despliega en pantalla la línea donde ocurrió el error y el texto relativo al error en base al siguiente arreglo de errores :

```
/* ----------------------------mensajes de error---------------------------------- */
char *ers[] = { "nombre invalido",                    /*  1 */
                "longitud invalida",                  /*  2 */
                "falta una coma",                     /*  3 */
                "tipo de dato invalido",              /*  4 */
                "faltan las comillas"                 /*  5 */
                "comando #diccionario faltante",      /*  6 */
                "comando '#' faltante",               /*  7 */
                "fin de archivo inesperado",          /*  8 */
                "nombre de archivo duplicado",        /*  9 */
                "dato elemental desconocido",         /* 10 */
```

```
        "demasiados datos elementales",          /* 11 */
        "falta memoria",                         /* 12 */
        "nombre de archivo desconocido",         /* 13 */
        "demasiados índices por archivo",        /* 14 */
        "demasiados elementos por índice",       /* 15 */
        "dato elemental duplicado",              /* 16 */
        "demasiados archivos",                   /* 17 */
        "switch invalido"                        /* 18 */
};
```

- static void get_line(): Obtiene una línea del archivo de entrada siempre y cuando la línea de entrada no comience con ";" ya que se considera un comentario. Si no encuentra ninguna línea por leer marca error(8).

- static char *get_word(): Lee los caracteres de entrada hasta que encuentra una coma "," y retoma una apuntador a la cadena de caracteres leída.

- static void skip_white(char* s): Dada una cadena de caracteres de entrada mueve el apuntador a la cadena hasta que encuentra el primer caracter diferente de blanco.

- static void expect_comma(char **cp): Valida que el siguiente caracter en la cadena sea una coma "," sino marca error(3)

La función principal del programa recibe el nombre del archivo de entrada del diccionario que tiene una terminación "ddl" y un switch que el proceso que debe ejecutar. El programa lee una linea del archivo de entrada diferente de comentario y lo compara con la directiva #diccionario la cual indica el inicio del diccionario de atributos si localiza dicha cadena llama a la función de_dict() cuya función es la de parsing de todo el diccionario de atributos, sino localiza dicha cadena marca error(6) y error(7) y lee la siguiente línea. Si el programa recibe el switch -4 llama a la función crea_dict() la cual se encarga de generar el diccionario de atributos con terminación "dic" que servirá para todo el proceso de diseño y construcción de la base de datos, si no encuentra dicho switch marca error(18).y termina el programa.

```
/* ------------------------- main program ------------------------- */
main(argc, argv)
int argc;
char *argv[];
{
   get_line();
        if (strncmp(ln, "#diccionario ", 12) == 0)
                de_dict();
        else  {
                error(6);
                error(7);
        get_line();
   }
   if (argc > 1)  {
                if (strcmp(argv[1], "-4") == 0)
                        crea_dic();
                else
                        error(18);
   }
   else
                error(18);
                depart(0);
}
```

La función principal de_dict() se encarga de controlar la gramática de construcción del diccionario prácticamente se encuentra en un while infinito el cual rompe hasta que a través de la función get_line() encuentra la cadena **#fin diccionario** o la función encuentra un fin de archivo inesperado y marca error(8).

Con el apuntador a la línea leída obtiene la primera palabra que corresponde al nombre del atributo, checa que sea un nombre valido y que no exista previamente en el diccionario, si no existe error almacena el nombre del atributo en al estructura del diccionario (strcpy(dc[dectr].dename, word)), valida que el siguiente caracter sea una coma "," y salta todos los blancos que existan en la cadena. El caracter siguiente deberá corresponder al tipo del atributo, por lo tanto valida que pertenezca a (A, Z, C, N ó D) si no pertenece marca error(4). si es correcto almacena el atributo en la estructura del diccionario (dc[dectr].detype = *cp++;). valida que el siguiente caracter sea una coma "," y obtiene la siguiente palabra hasta que encuentra una coma "," valida que sea un número y si es correcto lo almacena en la estructura del diccionario (dc[dectr].delen = atoi(word);) valida que el siguiente caracter sea una coma, salta los blancos necesarios y valida que la mascara de captura del atributo esté delimitada por comillas ' " ' y si es correcto lo almacena en la estructura de datos del diccionario (strcpy(dc[dectr].demask, cp);)

Repite el procedimiento anterior hasta que encuentra el final del diccionario o el final del archivo.

```
/* -------- construcción del diccionario de datos ---------- */
static void de_dict()
{
   char *cp, *cp1;
   int el;
   while (TRUE)   {
          get_line();
          if (strncmp(ln, "#fin diccionario", 16) == 0)
             break;
          if (dectr == MXELE) {
             error(11);
             continue;
          }
          cp = get_word(ln);
          name_val();
          for (el = 0; el < dectr; el++)
             if (strcmp(word, dc[el].dename) == 0)  {
                    error(16);
                    continue;
             }
          strcpy(dc[dectr].dename, word);
          expect_comma(&cp);
          skip_white(&cp);
          switch (*cp)  {
             case 'A':
             case 'Z':
             case 'C':
             case 'N':
             case 'D':  break;
             default :  error(4);
                              continue;
          }
          dc[dectr].detype = *cp++;
          expect_comma(&cp);
          cp = get_word(cp);
```

```
            numb_val();
            dc[dectr].delen = atoi(word);
            expect_comma(&cp);
            skip_white(&cp);
            if (*cp != '"') {
               error(5);
               continue;
            }
            cp1 = cp + 1;
            while (*cp1 != '"' && *cp1 && *cp1 != '\n')
               cp1++;
            if (*cp1++ != '"') {
               error(5);
               continue;
            }
            *cp1 = '\0';
            if ((dc[dectr].demask = malloc((cp1-cp)+1)) == 0)  {
               error(12);
               depart (1);
            }
            strcpy(dc[dectr].demask, cp);
            dectr++;
   }
}
```

La función crea_dic() es una función sencilla que se encarga de imprimir el diccionario previamente creado en la estructura **dc**, tal como se muestra a continuación. :

```
/*----------creación de diccionario_____-*/

void crea_dic()
{
        int el;

        for(el=0; el<dectr; el++)
        {
        printf("%s,", dc[el].dename);
        printf("%c,", dc[el].detype);
        printf("%d,", dc[el].delen);
        printf("%s\n", dc[el].demask);
        }
}
```

4. DICCIONARIO Y EDITOR DE ICONOS EVE (DEI)

El diccionario de iconos está formado por los constructores del modelo Entidad-Vínculo-Extendido de Chen. El diccionario esta constituido por dos partes: una parte que incluye las descripciones gráficas de los iconos que manipulará el usuario y otra parte que incluye la descripción de los parámetros y las estructuras básicas para el control del intérprete. Los iconos de la figura siguiente representan el diccionario empleado en la gramática de construcción, en el momento de la implantación de SDBD se han eliminado algunos iconos del diccionario para optimizar la construcción de diagramas EVE, que corresponde a los referentes a los atributos : los iconos CI y CID, ya que en diseño definitivo del editor de iconos cada icono tiene asociadas propiedades, entre ellas el nombre de la entidad y dos listas de atributos una que corresponde a los atributos llave y otra a los atributos descriptores. La decisión anterior se tomo debido a que simplificaba la edición de diagramas de Chen.

El editor de iconos EVE cumple la función de proporcionar una herramienta de edición que con base en el diccionario de iconos, que se muestra en la figura 1, permita la edición completa de diagramas de Chen incluyendo funciones para salvar y recuperar diagramas editados, así como funciones básicas para relacionar entidades a través de vínculos y constructores de generalización y especialización. Así también dicha herramienta realiza una edición dirigida por sintaxis, ya que durante la etapa de edición valida que cumpla ciertas reglas de sintaxis expresadas por la gramática de construcción GC que se describe en el capitulo 5 y aquellas reglas que no se puedan validar durante la edición son checadas por un proceso de parsing, de tal manera que valida los siguientes errores :

- --"Entidad sin relacionar"
- --"Conector de generalización sin entradas definidas"
- --"Conector de generalización sin salidas definidas"
- --"Conector de especialización sin entradas definidas"
- --"Conector de especialización sin salidas definidas"
- --"Conector con número de entradas incompletas"
- --"Atributo: indefinido"
- --"Atributo indefinido","
- --"No es valida una conexión entre entidades"
- --"El icono generalizado/especializado debe ser entidad generalizada"
- --"No es valida una doble conexión al mismo icono"
- --"Un icono conector unario no puede tener mas de un icono asociado"
- --"Un icono conector binario no puede tener mas de dos iconos asociados"
- --"Un icono conector ternario no puede tener mas de tres iconos asociados"
- --"Un icono generalización/especialización no puede ir hacia mas de un icono"
- --"No es valida la generalización/especialización anidada"
- --"No existe conexión entre estos iconos"

Así también el Editor de Iconos EVE incluye utilerías para el manejo de la escala del diagrama editado y funciones para scroll dentro de un diagrama. En el capitulo 8 se explica con detalle el funcionamiento de dicho editor.

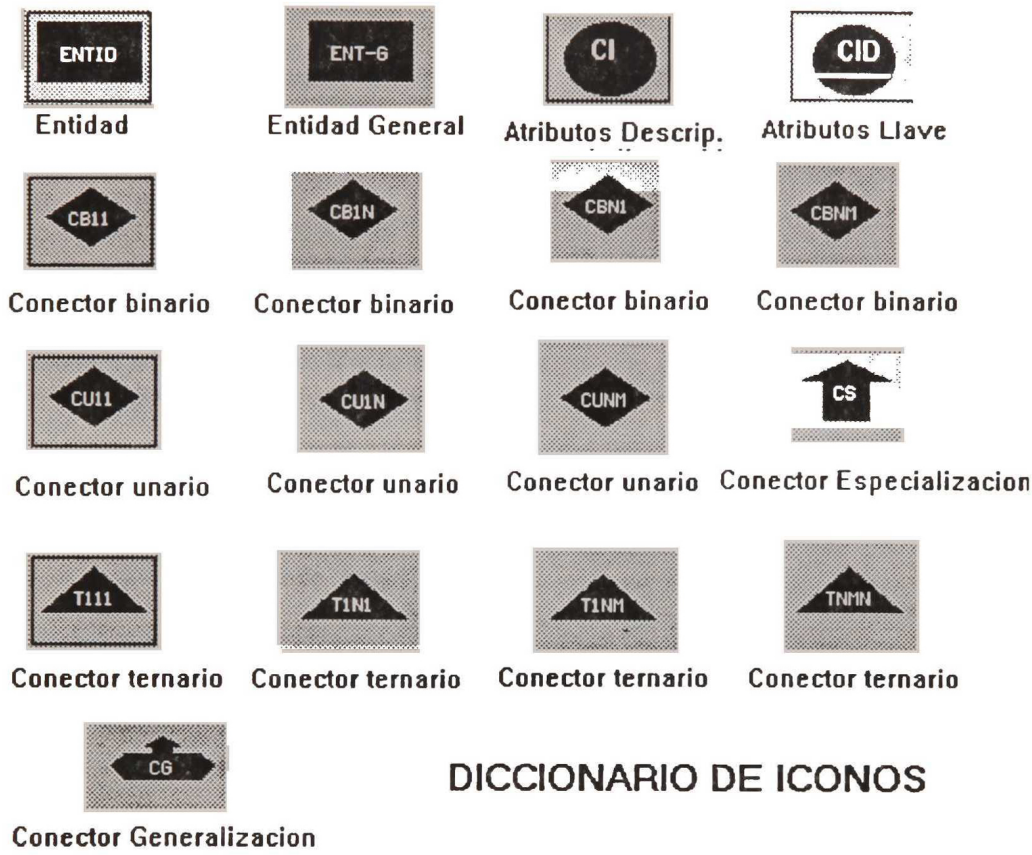


Figura 1: Diccionario de Iconos

4.1. Anotaciones sobre la implementación del editor de Iconos en lenguaje C

A continuación se describe las principales características del editor de diagramas Entidad-Vinculo Extendido de Chen. Este módulo nos permite crear diagramas usando un conjunto de iconos relacionados unos con otros a través de lineas, teniendo al final un diagrama que describe conceptualmente una Base de Datos. El editor es un programa bastante amplio, por tal motivo solo se presentan las estructuras de datos utilizadas en lenguaje C y se describen las funciones generales de su funcionamiento.

El editor utiliza como estructura principal una retícula rectangular que juega el papel de espacio de trabajo. Esta retícula se llama "matriz" y sus dimensiones son 34 X 26. Los elementos que contiene esta matriz están formados por elementos de la siguiente estructura:

```
esticono matriz[34][26];
```

y la estructura **esticono** esta definida de la siguiente manera :

```
typedef struct esticono_{
        int tipo;
        char nombre[20];
        textoap llave;
        textoap descriptores;
        ligasap salen;
        ligasap entran;
       }esticono, *iconoap;
```

- **tipo** : Nos indica que elemento es el que actualmente ocupa un lugar en el espacio de trabajo, pudiendo ser una entidad o un conector.

- **nombre** : Permite almacenar el nombre del elemento.

- **llave** : Contiene información sobre una lista (ligada) que almacena los atributos que integran la llave para el elemento actual. El elemento llave de la estructura esticono es un apuntador a una lista ligada que tiene la estructura texto tal como se muestra a continuación:

```
typedef struct texto_  {
        char line[20];
        int activo;
        struct texto_ *next;
       } texto, *textoap;
```

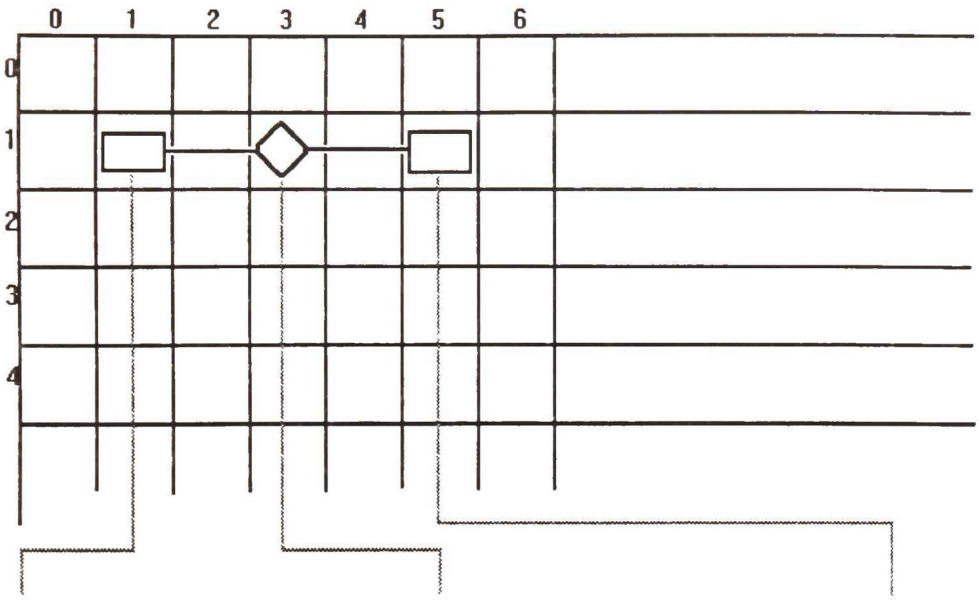
- **descriptores** : Juega el mismo papel que el elemento anterior, pero con los descriptores.

- **salen** : Es una lista de coordenadas (ligada) que tiene los datos para saber si del elemento actual salen lineas y hacia a donde. Esta información es importante debido a que facilita leer la semántica del diagrama. La estructura de dicha lista ligada se muestra a continuación

```
typedef struct ligas_{
    int coordx;
    int coordy;
    int activo;
    struct ligas_ *next;
   }ligas, *ligasap;
```

- **entran** : Tiene la misma estructura que el dato anterior, pero indica si al elemento actual entran lineas y de donde, la utilidad es la misma.

Para entender mas estos términos, veamos y estudiemos la estructura interna del siguiente diagrama:



Tipo: 0	Tipo: 1	Tipo: 0
Nombre :Clientes	Nombre : Cuenta	Nombre : Bancos
LLave : No_Cuenta		LLave : Nombre_Banco
Descriptores : ...		Descriptores:
Salen : (3,1)	Salen : NULL	Salen : (3,1)
Entran : NULL	Entran : (1,1) , (5,1)	Entran : NULL

Toda esta información es almacenada en la estructura por algunas funciones, en seguida describiremos las mas importantes.

- **edit_diagram1** .- Es la función principal de edición, se basa en un ciclo de lectura (que finaliza presionando ESC) y actúa dependiendo de la tecla o teclas que se presionen.

  Los pasos que sigue son:
  1. Dibuja el menú principal y regiones de trabajo.
  2. Dibuja las lineas que relacionan a unos iconos con otros.
  3. Dibuja los iconos.
  4. Entra al ciclo de operación.

  Dentro del ciclo de operación, el usuario puede realizar las siguientes tareas:

  • **Desplazarse** por toda la retícula de trabajo usando las teclas de desplazamiento, el editor soporta "scroll" vertical y horizontal.

  • **Borrar un icono**. Elimina el icono actualmente seleccionado por el cursor de la retícula (pone al valor -1) y libera las relaciones de este con otros iconos y los otros iconos con el que se eliminó.

  • **Conectar a otro icono**. Permite relacionar al icono actualmente seleccionado con otros elementos de un diagrama. Por ejemplo, conectar a una entidad con un vínculo. Las conexiones parten de iconos entidad y llegan a vínculos o categorías de especialización o generalización. O parten de las categorías de especialización o generalización y llegan a entidades de generalización. Esta operación realiza la actualización en los elementos de la estructura correspondiente (ENTRAN y SALEN).

- **Cambio de la escala de edición**. Permite alterar la escala de visualización de trazo de los iconos. Las escalas disponibles son: 1:1, 1:2, 1:4

- **Insertar un icono**. Permite colocar un icono en la retícula de edición, el espacio en esta retícula se llena con la información del icono que el usuario haya seleccionado.

- **Editar los atributos** llaves o descriptores, o el nombre del icono. Nos permite introducir elementos en la estructura que contiene cada celda de la retícula, específicamente en la parte LLAVES y DESCRIPTORES o NOMBRE. La lectura se hace por linea y finaliza cuando se lee una linea en blanco.

- **Agregar atributos llave**. Nos permite agregar a la lista ligada de LLAVES un nuevo elemento, verificando que no exista ya en la lista.

- **Borrar atributos llave.** Nos permite eliminar de la lista ligada de LLAVES un elemento.

- **Agregar atributos descriptores**. Nos permite agregar a la lista de DESCRIPTORES un nuevo elemento, verificando que no exista ya en la lista.

- **Borrar atributos descriptores**. Nos permite eliminar de la lista ligada de DESCRIPTORES un elemento.

- **Borrar una conexión**. Nos permite eliminar un arco que une a dos elementos gráficos, la forma de borrado es posicionar el cursor en el elemento del que sale el arco, en seguida se invoca esta operación y se selecciona el icono a donde llega la conexión, en este punto de presiona <ENTER>. La información de este arco es eliminada de la parte "SALEN" y/o "ENTRAN" de los iconos involucrados.

- **Lectura de un nuevo diagrama**. Permite recuperar de disco un diagrama. Si previamente existía uno cargado en el editor, se libera la memoria que ocupa y se inicializa la retícula de trabajo.

- **Salva a disco el diagrama actual**. Esta función almacena en disco la información necesaria para posteriormente recuperar un diagrama.

- **Análisis del diagrama**. Esta opción realiza una verificación sobre la construcción del diagrama, verificando entre otras cosas, que no existan elementos aislados, atributos en el diagrama que no estén definidos en el diccionario de datos del mismo, etc. Si existe algún error, el modulo nos avisa indicando en que posición se encuentra el icono o la información invalida.

Una vez finalizada alguna de las operaciones anteriores, se realiza una actualización de la información actualmente desplegada.

La estructura de los archivos generados por el editor es la siguiente:

m-número de iconos del diagrama.

.
.

Para cada icono i del diagrama se almacena la siguiente información:

x - coordenada en x que ocupa el icono i en la retícula.
y - coordenada en y que ocupa el icono i en la retícula.
t - Tipo del icono i.
n - Nombre del icono i.
l - número de atributos llave.
ll - Lista de atributos llave (uno por linea - upl).

d - número de atributos descriptores.
dd - Lista de atributos descriptores (uno por linea - upl).
a - número de arcos que entran al icono i.
aa - Lista de pares ordenados de arcos (v,w) que indican el origen de los mismos (upl).
b - número de arcos que salen del icono i.
bb - Lista de pares ordenados de arcos (v,w) que indican el destino de los mismos (upl).

5. GRAMATICA DE CONSTRUCCION (GC)

La Gramática de Construcción determina la sintaxis para la construcción de los diagramas EVE y delimita la manera en que los elementos del diccionario de iconos pueden ser agrupados en la construcción. Su utilización dentro del proyecto es restringido, ya que la validación sintáctica de un diagrama de Chen se realiza en dos partes : La primera durante la edición de un diagrama de Chen se checa la sintaxis de construcción y durante el Parsing del Diagrama se checa que cumpla las reglas sintácticas que se expresan en la gramática siguiente :

Gramática de Construcción de Diagramas EVE.

<**Diagrama**> ::= <sentidad> | { <relación > & <relación> }
<sentidad> ::= <entidad> | <jerarquía>
<entidad>::=(rectángulo(<identificador>) - círculoid(<atributos>)) | (rectángulo(<identificador>)-circuloid(<atributos>), circulo(<atributos>))
<entidadc> ::= (recdoble(<identificador>) - círculoid(<atributos>)) | (recdoble(<identificador>) - círculoid(<atributos>), círculo(<atributos>))
<conector> ::= <ct> | <cb> | <cu> | <cg> | <cs>
<ct> ::= <ct111> | <ct11n> | <ct1mn> | <ctmnm>
<cb> ::= <cb11> | <cb1n> | <cbmn>
<cu> ::= <cu11> | <cu1n> | <cumn>
<relación> ::= <rt> | <rb> | <ru>
<rt> ::= [sentidad1 <ct> sentidad2 , sentidad3]
<rb> ::= [sentidad1 <cb> sentidad2]
<ru> ::= [sentidad <cu>]
<entidadg> ::= <diagrama> | (* <diagrama> + <diagrama> *)
<generalización> ::= [* <entidadc> <cg> <entidadg> *]
<entidads> ::= <diagrama> | (* <diagrama> % <diagrama> *)
<subconjunto> ::= [* <entidadc> <cs> <entidads> *]
<jeraquía> ::= <subconjunto> | <generalización>
<atributos> ::= <natributos> <natributoss>
<natributos> ::= <identificador> [, <identificador]*
<natributoss> ::= <natributos> <natributoss> | ε
<identificador> ::= <letra> <caracteres>
<caracteres> ::= <caracter> <caracteres> | ε
<tipo> ::= A | D | N | Z | C
<letra> ::= a | b | c | ... | z | A | B | C | ... | Z
<número> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<caracter> ::= <letra> | <número> | _ | | / | (|) | * | $ | % | & | @

Por ejemplo, suponga el siguiente diagrama entidad-vínculo de la figura 2:

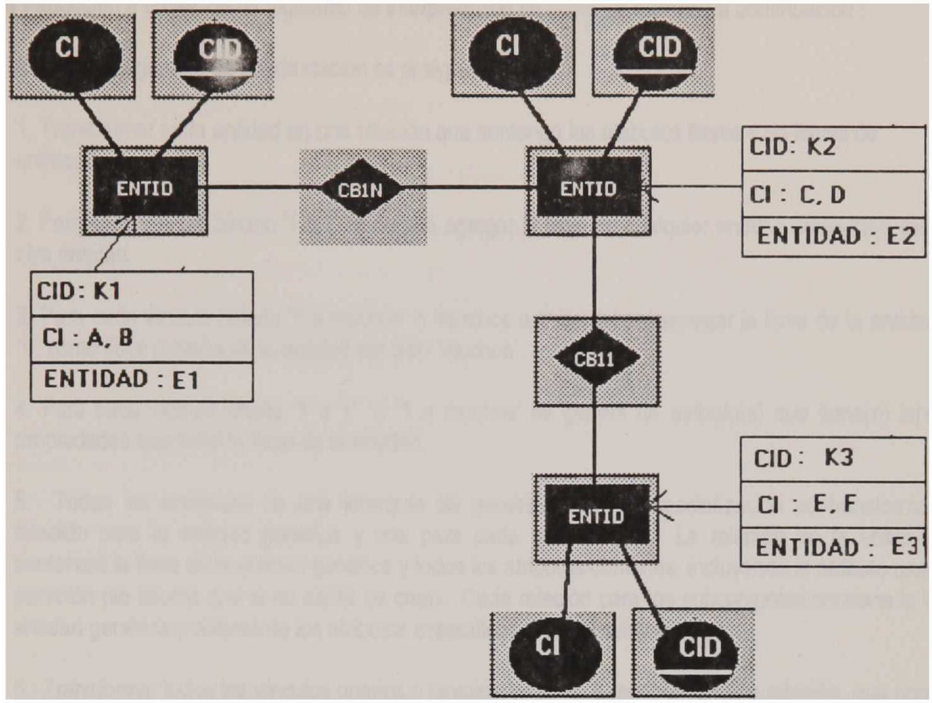


Figura 2: Diagrama Entidad-Vínculo de Prueba

Lo cual puede ser descrito sintácticamente a partir de :

Definiciones auxiliares :

Entidad1 ::= (rectángulo(E1) - círculoid(K1), círculo(A, B))
Entidad2 ::= (rectángulo(E2) - círculoid(K2), círculo(C, D))
Entidad3 ::= (rectángulo(E3) - círculoid(K3), círculo(E, F))

DIAGRAMA PRUEBA::= { [Entidad1 <cb1m> Entidad2] & [Entidad2 <cb11> Entidad3] }

Como se mencionó con anterioridad en la implementación de SDBD no se incluyeron los iconos para atributos, es decir los iconos CI y CID, ya que en diseño definitivo del editor de iconos cada icono tiene asociadas propiedades, entre ellas el nombre de la entidad y dos listas de atributos una que corresponde a los atributos llave y otra a los atributos descriptores. La decisión anterior se tomó debido a que simplificaba la edición de diagramas de Chen.

Por último cabe señalar que uso de la gramática se restringió a la validación sintáctica de los diagramas de Chen, aunque con ligeras modificaciones en la implementación, ya que era más sencillo realizar una edición de diagramas de Chen dirigidos por sintaxis, es decir, durante la etapa de edición validar que cumpla ciertas reglas de sintaxis expresadas por la gramática y aquellas reglas que no se puedan validar durante la edición son checadas por un proceso de Parsing.

6. INTERPRETE (INT)

El intérprete se encarga de la transformación de dicho diagrama EVE, sintácticamente bien construido, al modelo relacional de Codd. Inicialmente se pensó realizar dicha transformación dirigida por sintaxis usando la gramática de construcción previamente descrita **GC** pero debido a anteriores experiencias con LIDA se decidió

realizar la traducción a través de un algoritmo de interpretación tal como se describe a continuación :

El algoritmo general de interpretación es el siguiente :

1. Transformar cada entidad en una relación que contenga los atributos llaves y no-llaves de la entidad.

2. Para cada vínculo binario '1 a 1' se deberá agregar la llave de cualquier entidad como llave extraña en la otra entidad.

3. Para cada vínculo binario '1 a muchos' o 'muchos a 1' se deberá agregar la llave de la entidad del lado '1', como llave extraña en la entidad del lado 'muchos'.

4. Para cada vínculo unario '1 a 1' o '1 a muchos' se genera un atributo(s) que tiene(n) la(s) mismas propiedades que tiene la llave de la entidad.

5. Todas las entidades de una jerarquía de generalización o especialización se transforman en una relación para la entidad genérica y una para cada subconjunto. La relación de la entidad genérica contendrá la llave de la entidad genérica y todos los atributos comunes, incluyendo el atributo usado para la partición (se asume que si no existe se crea). Cada relación para los subconjuntos contiene la llave de la entidad genérica y solamente los atributos específicos para el subconjunto.

6. Transformar todos los vínculos unarios o binarios 'muchos a muchos' en una relación, que contendrá los atributos llaves de cada entidad participante en el vínculo.

7. Transformar todos los vínculos ternarios en una relación que contendrá los atributos llave de cada entidad participante en el vínculo. Dependiendo del grado del vínculo ternario se producen diferentes dependencias funcionales entre los atributos de la relación resultante.

El interprete se encarga de la traducción de un diagrama de Chen a un modelo relacional de Codd sin normalizar, por lo tanto el modelo relacional de Codd deberá ser normalizado para obtener relaciones en la tercera forma normal. Para una explicación más detallada del modelo de Codd consulte [HERN90], aquí únicamente expondremos la forma en como se representan las relaciones de Codd.

Dada la siguiente relación :

Relación : PROMOTORES
Llaves : DIGITO, NUMAPO
Descriptores : FECING, NOMBRE, REGLEG, RFC, NUMSUC, DIGITO_1,NUMAPO_1

Para mostrarla en notación de Codd se expresa como un vector donde el nombre de la entidad corresponde al nombre del vector, los atributos de la relación se expresan como elementos del vector pero con la condicionante que los atributos Llaves se muestran con subraya. Por último todas aquellas llaves extrañas se muestran en letras itálicas. Así la anterior entidad podrá ser expresada como :

PROMOTORES(DIGITO,NUMAPO,FECING,NOMBRE,REGLEG,RFC,*NUMSUC*,DIGITO_1,NUMAPO_1)

Aunque la notación vectorial es muy comúnmente manejada para expresar las relaciones, no es muy manejable desde el aspecto de programación por lo tanto dentro de SDBD se utilizó una gramática propia para expresar relaciones, la cual se muestra a continuación :

<Relaciones de Codd> ::= <relaciones> <alias>
<relaciones> ::= <nrelacion> <nrelaciones>

<nrelacion> ::= #name_rel <identificador> <llaves> <descriptores>
<nrelaciones> ::= <nrelacion> <nrelaciones> | ε
<llaves> ::= <nllave> <nllaves>
<nllave> ::= #keys <atributosllave> #endkeys
<nllaves> ::= <nllave> <nllaves> | ε
<atributosllave> ::= <natributosllave> <natributosllaves>
<natributosllave> ::= <identificador>
<natributosllaves> ::= <natributosllave> <natributosllaves> | ε
<descriptores> ::= <ndescriptores> <ndescriptoress>
<ndescriptores> ::= #desc <atributosdescriptores> #enddesc
<ndescriptoress> ::= <ndescriptores> <ndescriptoress> | ε
<atributosdescriptores> ::= <natributosdescriptores> <natributosdescriptoress>
<natributosdescriptores> ::= <identificador>
<natributosdescriptoress> ::= <natributosdescriptores> <natributosdescriptoress> | ε
<alias> ::= <nalias> <naliass>
<nalias> ::= #gene <identificador> #fuente <identificador>
<naliass> ::= <nalias> <naliass> | ε
<identificador> ::= <letra> <caracteres>
<caracteres> ::= <caracter> <caracteres> | ε
<letra> ::= a | b | c | ... | z | A | B | C | ... | Z
<número> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<caracter> ::= <letra> | <número> | _ | | / | (|) | * | $ | % | & | @

De tal forma que la relación PROMOTORES queda expresada de la siguiente forma :

#name_rel
PROMOTORES
#keys
DIGITO
NUMAPO
#endkeys
#desc
FECING
NOMBRE
REGLEG
RFC
NUMSUC
DIGITO_1
NUMAPO_1
#enddesc
#gene
DIGITO_1
#fuente
DIGITO
#gene
NUMAPO_1
#fuente
NUMAPO

En el Anexo 4 donde se muestra la traducción definitiva del SAPF puede notarse el uso extensivo de esta nomenclatura.

6.1. Anotaciones sobre la implementación del Intérprete (Traductor)

A continuación se listan cada uno de los pasos que se siguen durante este proceso. Debido a que el procesos es complejo y la programación tiene un uso amplio de manejo de listas para describir programación de este proceso se decidió utilizar una notación especial que se muestra a continuación :

- Se usa una lista de resultados L , llamada en el texto lista de traducción que cuenta con los siguientes datos para cada elemento de ella:

```
typedef struct list_traduc{
        char nombre[20];
        textoap llave;
        textoap descriptores;
        int contador;
        struct list_traduc *next;
       }*aplist_tr,list_tr;
```

- Se usa una lista de atributos compatibles L1. Cada elemento de la estructura tiene los siguientes elementos :

```
typedef struct compat_ {
        char generado[20];
        char fuente[20];
        struct compat_ *next;
        }compat, *apcomp;
```

- Las funciones que son utilizadas de forma común son las siguientes :

- - para accesar elementos de una estructura del diagrama se usa "**estructura**"."**elemento**".
- - **llaves(X)**:obtiene las llaves del elemento X dentro de L.
- - **descriptores(X)**:obtiene las llaves del elemento X dentro de L.
- - **nombre(X)**:obtiene el nombre del elemento X dentro de L.
- - **elemento_generado(X)**:obtiene el atributo generado del elemento X dentro de L1.
- - **atributo_base(X)**:obtiene el atributo base del elemento X dentro de L1.
- - **union(X,Y)**:obtiene la union de las listas de atributos X y Y.
- - **interseccion(X,Y)**:obtiene la intersección de las listas de atributos X y Y.
- - **diferencia(X,Y)**:obtiene la diferencia de las listas de atributos X y Y.
- - **<nombre>**:nombre de la función que se aplica.

En seguida se listan las funciones de las fases de traducción:

1. **<gen_entidad>**. Para todo icono X que sea de tipo "entidad" y que no este en la lista de traducción L, se agrega a la misma con los datos (X.nombre,X.llaves,X.descriptores):

2. **<gen_conbin11>**. Para todo icono X que sea de tipo "conector binario 1 a 1":
2.a. Recupera los elementos X1 y Y1 que se relacionan a través del icono X de la forma X1--->X<---Y1.
2.b. Se obtiene z=union(descriptores(X1),llaves(Y1)).
2.c. Se busca el elemento X1 en la lista de traducción L y se hace descriptores(X1)=z.

3. **<gen_conbin1m>**. Para todo icono X que sea de tipo "conector binario 1 a m":
3.a. Se recuperan los elementos X1 y Y1 que estan relacionados a través de X de la forma X1--->X<---Y1.
3.b. Se obtiene z=union(descriptores(Y1),llaves(X1)).

3.c. Se busca el elemento Y1 en la lista de traducción L y se hace descriptores(Y1)=z.

4. **<gen_conbinm1>**. Para todo icono X que sea de tipo "conector binario m a 1":
4.a. Se recuperan los elementos X1 y Y1 que estan relacionados a través de X de la forma X1--->X<---Y1.
4.b. Se obtiene z=union(descriptores(X1),llaves(Y1)).
4.c. Se busca el elemento X1 en la lista de traducción L y se hace descriptores(X1)=z.

5. **<gen_conuna111m>**. Para todo icono X que sea del tipo "conector unario 1 a 1, 1 a m":
5.a. Se recupera el elemento X1 que esta relacionado con X de la forma X<--->X1.
5.b. Se genera un conjunto de atributos "a" tomando como base los atributos en descriptores(X1). Se actualiza la lista L1 con los pares ordenados de atributos correspondientes.
5.c. Se obtiene z=union(descriptores(X1,a).
5.d. Se busca el elemento X1 en la lista de traducción y se hace descriptores(X1)=z.

6. **<gen_congenesp>**. Para todo icono X que sea de tipo "generalización" o "especialización":
6.a. Se obtiene el icono X1 que se esta generalizando o especializando.
6.b. Se obtienen los iconos Y1,Y2, ... Yn que participan en la generalización o especialización.
6.c. Se obtiene z=interseccion(llaves(Y1),llaves(Y2),...,llaves(Yn)).
6.d. Se obtiene z1=interseccion(descriptores(Y1),...,descriptores(Yn)).
6.e. Si el elemento X1 no se encuentra en la lista de traducción L, entonces se agrega con los datos (X1.nombre,z1,z).
6.f. Si el elemento X1 si se encuentra en la lista de traducción L, entonces se busca y se hace llaves(X1)=z, descriptores(X1)=z1.
6.g. Para cada elemento Yi (i=1 a n) se hace descriptores(Yi)=diferencia(descriptores(Yi),z1).

7. **<gen_conbinnm>**. Para todo icono X que sea del tipo "conector binario n a m":
7.a. Se recuperan los elementos X1 y Y1 que estan relacionados a través de X de la forma X1--->X<---Y1.
7.b. Se obtiene z=union(llaves(X1),llaves(Y1)).
7.c. Se agrega un nuevo elemento a la lista de traducción L con los datos (X.nombre,z,NIL). NIL representa el conjunto vacío.

8. **<gen_conunanm>**. Para todo icono X que sea de tipo "conector unario n a m":
8.a. Se obtiene el elemento X1 que esta relacionado con el icono X de la forma X<--->X1.
8.b. Se genera un conjunto de atributos "a" tomando como base los atributos en llaves(X1). Se agregan los elementos necesarios en la lista L1.
8.c. Se obtiene z=union(llaves(X1),a).
8.d. Se agrega un nuevo elemento a la lista de traducción L con los datos (X.nombre,z,NIL). NIL representa una lista vacía.

9. **<gen_conter>**. Para todo icono X que sea de tipo "conector ternario":
9.a. Se recuperan los elementos X1,Y1 y Z1 que estan relacionados a través del icono X.
9.b. Se obtiene z=union(llaves(X1),llaves(Y1),llaves(Z1)).
9.c. Se agrega un nuevo elemento a la lista de traducción L con los datos (X.nombre,z,NIL). NIL representa la lista vacía.

10. <genera_result>. Genera el archivo de resultados <nombre>.rel usando L. Para cada elemento X en L, se escribe en el archivo de la siguiente manera:

```
#name_rel
nombre(X)
#keys
llaves(X)          /* Una por linea */
#endkeys
```

```
#desc
descriptores(X)   /* Uno por linea */
#enddesc
```

11. <genera_compat>. Genera la lista de atributos compatibles usando la lista L1. Para cada elemento X1 en L1 se escribe:

```
#gene
elemento_generado(X1)
#fuente
elemento_fuente(X1)
```

7. PROCESOS (PRO)

Los procesos básicos del sistemas son el de **Normalización** y el de **Diseño Físico**. La normalización se llevará acabo a través del algoritmo de Bernstein y el proceso de diseño físico se realizará a través del manejador de bases de datos utilizado para LIDA.

7.1. NORMALIZACION

La normalización es comúnmente preservada bajo las transformaciones anteriores del INTERPRETE, pero la introducción de llaves extrañas puede producir dependencias adicionales que causen alguna desnormalización. Así que después de la transformación la normalización se restablecerá usando el algoritmo de Bernstein [BERN76].

La normalización tiene que ver con el proceso de diseño del esquema de relaciones de la base de datos y tiene por objetivo el mantener la integridad de la base de datos a través de minimizar la redundancia e inconsistencia dentro de las relaciones. Beeri [BEER78] en su trabajo sobre la teoría de la normalización de base de datos señala que existen dos aproximaciones al problema de normalización, síntesis y descomposición. El algoritmo más conocido para la aproximación de síntesis es el de Bernstein [BERN76] que se basa en la síntesis de relaciones en tercera forma normal a partir de un conjunto de dependencias funcionales. Por el lado de la descomposición los algoritmos mas conocidos son el de Fagin [FAGI77] y el de Rissanen [RISS77], obteniendo descomposiciones en tercera y cuarta forma normal. Aquí no se expondrá la aproximación de descomposición ni la teoría general de normalización, si se desea profundizar en estos conceptos consulte el trabajo de Hernández [HERN90].

El primer problema al que nos enfrentamos en la implementación del algoritmo de Bernstein es la manipulación del conjunto de entrada de dependencias funcionales. Dado un conjunto de dependencias funcionales en una relación es posible deducir otras dependencias aplicando un conjunto de reglas llamadas axiomas de Amstrong [BEER78] :

1. Reflexividad : Si A y B son un conjunto de atributos de la relación R donde $B \subseteq A$, entonces A --> B.

2. Aumentación : Si A, B y C son conjuntos de atributos de la relación R y A --> B entonces se tiene que AC --> BC

3. Transitividad : Si A, B y C son conjuntos de atributos de la relación R y A --> B, B --> C, entonces se tiene que A --> C.

Al conjunto de todas las dependencias funcionales que siguen lógicamente del conjunto de entrada aplicando los axiomas de Amstrong se le llama la "cerradura del conjunto de dependencias". El procesos para la obtención de dicha cerradura es muy largo y no es muy útil, ya que el primer axioma genera las llamadas "dependencias triviales". Una manera de simplificar este conjunto y eliminar fuentes de redundancia es construir la cerradura del conjunto de dependencias a partir de una cubierta minimal. Una cubierta minimal del conjunto de dependencias de entrada, es un nuevo conjunto de dependencias que generan la misma cerradura del conjunto de entrada, es decir, que contiene la misma información potencial, además de que los lados derechos de las

dependencias funcionales son conjuntos de un solo elemento, no existen dependencias funcionales en la cubierta que puedan ser deducidas de otras y los lados izquierdos de las dependencias funcionales no contienen atributos extraños.

El algoritmo de Bernstein se basa en el hecho que las dependencias funcionales son incorporadas en las llaves de las relaciones sintetizadas y que el conjunto de relaciones obtenido satisface la propiedad de ser mínimo. El paso inicial de dicho algoritmo es la obtención de una cubierta minimal, para ello, primeramente se eliminan los atributos extraños y en seguida se eliminan las dependencias funcionales redundantes. Para una exposición más detallada del algoritmo consulte [HERN90].

La cerradura de un conjunto de atributos X con respecto a un conjunto de dependencias funcionales F, el cual puede ser denotado como X*. es el conjunto de todos los atributos derivados de X usando las dependencias funcionales F y los axiomas de Armstrong [DIED88]. Este conjunto es construido recursivamente, buscando en cada nivel una dependencia funcional tal que su lado izquierdo (LHS) sea un subconjunto de X. Cuando una dependencia de este tipo se encuentra, el lado derecho de la dependencia (RHS) es anexado al cerradura y el algoritmo se llama recursivamente con un nuevo conjunto construido uniendo X con RHS. Si tal dependencia no se encuentra la recursión termina.

ALGORITMO CERRADURA.
{Cerradura de X con respecto a F}

Entrada: un conjunto de dependencias F, un conjunto de atributos X.
Salida: CERRADURA, la cerradura de X con respecto a F.

1. Asigna CERRADURA = X.
2. Repetir hasta que no se adicione más atributos a CERRADURA:
   Para cada dependencia Y --> A en F.
      Si Y es subconjunto de CERRADURA y A no pertenece a CERRADURA
         Asigna CERRADURA = CERRADURA unión {A}

El primer paso para la construcción de la cubierta minimal es eliminar los atributos extraños de los lados izquierdos de las dependencias funcional (LHS), donde un atributo es extraño si puede ser eliminado del lado izquierdo de la dependencia sin cambiar la cerradura del conjunto de entrada de dependencias [MAIE83]. El algoritmo busca un atributo A en el lado izquierdo de la dependencia funcional, construye la diferencia entre el lado izquierdo y el atributo, evalúa la cerradura de dicha diferencia. Prueba si el lado derecho pertenece a esta cerradura, si esto sucede, el atributo A es extraño, y el algoritmo se llama recursivamente hasta que la reducción no pueda continuar.

ALGORITMO EXTRAÑOS.
{Eliminación de atributos extraños}.

Entrada: un conjunto de dependencias funcionales F.
Salida: Un conjunto de dependencias F' reducido por la izquierda.

0. Para cada dependencia X --> B en F.
   1. Asigna Z = X.
   2. Para cada atributo A pertenece a X
      Si B esta en (Z - A)*
         entonces asigna Z = (Z - A)

3. Reemplaza X --> B por Z --> B.

El segundo paso de la construcción de la cubierta minimal, es la eliminación de las dependencias funcionales redundantes. Una manera de determinar si una dependencia funcional es redundante o no, consiste en eliminarla del conjunto de entrada y calcular la cerradura del lado izquierdo de la dependencia con respecto del conjunto más pequeño de dependencias. Si el lado derecho de la dependencia está en esta cerradura, entonces la dependencia es implicada y puede ser eliminada.

ALGORITMO ELIMINA
{Eliminación de dependencias funcionales redundantes}

Entrada: Un conjunto de dependencias F, reducido por la izquierda.
Salida: Una cubierta minimal H para F.

1. Asigna H = F.
2. Para cada dependencia X --> A en F.
   Asigna G = H - (X --> A)
   Si A pertenece a X* con respecto a G
      entonces H = G.

Por último, una vez obtenida la cubierta minimal se puede ejecutar el algoritmo de Bernstein, que se puede expresar de la siguiente forma :

ALGORITMO BERNSTEIN
{Algoritmo de Bernstein}

Entrada: Un conjunto de dependencias funcional F.
Salida: Una colección de relaciones en tercera forma normal.

1. Encontrar una cubierta minimal H para F. Eliminando los atributos extraños de los lados izquierdos de las dependencias y las dependencias funcionales redundantes.

2. Hacer una partición del conjunto H en conjuntos Hi tal que todas las dependencias en cada grupo tengan lados izquierdos idénticos.

3. Unir dos grupos Hi y Hj siempre y cuando sus lados izquierdos sean llaves equivalentes, es decir, siempre que las dependencias Y --> X y X --> Y permanezcan.

4. Construir una cubierta particular de H que incluya todas las dependencias X --> Y, donde X y Y son llaves equivalentes, tal que todas las otras dependencias son no-redundantes. Eliminar las dependencias transitivas particulares.

5. Para cada grupo sintetizar una relación que consiste de los atributos que aparecen en cada grupo.

Por ejemplo :

Sea R una relación con el siguiente esquema R (A,B,C,D,E,F) donde se cumple el siguiente conjunto de dependencias :

1. (A, B) --> C

2. C --> A
3. D --> E
4. (D, E) --> F
5. E --> D
6. E --> F
7. E --> A.

Paso 1: Determinación de la cubierta minimal H.

√ Eliminación de atributos extraños : en la dependencia (D, E) --> F se elimina el atributo extraño D, quedando el lado izquierdo de la dependencia como (E), Note que una vez eliminado el atributo no cambia la cerradura de F.

√ Eliminación de dependencias funcionales redundantes : se eliminan la dependencia funcional 6 por ser redundante respecto de 4.

√ Así la cubierta minimal H esta formada por : (A, B) --> C; C --> A; D --> E; (E) --> F, E --> D; E --> A.

Paso 2: Partición de la cubierta H en grupos de lados izquierdos idénticos :

√ Grupo (A, B) --> (C)
√ Grupo (C) --> (A)
√ Grupo (D) --> (E)
√ Grupo (E) --> (D, A, F)

Paso 3: Uniendo grupos con llaves equivalentes : Como E y D son llaves equivalentes se forman los siguientes grupos

√ Grupo (A, B) --> (C)
√ Grupo (C) --> (A)
√ Grupo ((D), (E)) --> (D, A, F)

Paso 4: Se eliminan las dependencias transitivas y los grupos definidos son los siguientes :

√ Grupo (A, B) --> (C)
√ Grupo (C) --> (A)
√ Grupo ((D), (E)) --> (D, A, F)

Paso 5 : Síntesis de relaciones

√ R_a (A, B, C) Llaves (A, B)
√ R_b (A, C) Llaves (C)
√ R_c (A, D, E, F) Llaves (D), (E)

De igual manera que para el caso del interprete las relaciones normalizadas son expresadas en la notación de Codd y para el manejo dentro de SDBD se desarrollo se utilizó una gramática propia para expresar relaciones normalizadas, la cual se muestra a continuación :

<Relaciones Normalizadas> ::= <relaciones> <alias>
<relaciones> ::= <nrelacion> <nrelaciones>

<nrelacion> ::= #name_rel <identificador> <llaves> <descriptores>
<nrelaciones> ::= <nrelacion> <nrelaciones> | ε
<llaves> ::= <nllave> <nllaves>
<nllave> ::= #keys <atributosllave> #endkeys
<nllaves> ::= <nllave> <nllaves> | ε
<atributosllave> ::= <natributosllave> <natributosllaves>
<natributosllave> ::= <identificador>
<natributosllaves> ::= <natributosllave> <natributosllaves> | ε
<descriptores> ::= <ndescriptores> <ndescriptoress>
<ndescriptores> ::= #desc <atributosdescriptores> #enddesc
<ndescriptoress> ::= <ndescriptores> <ndescriptoress> | ε
<atributosdescriptores> ::= <natributosdescriptores> <natributosdescriptoress>
<natributosdescriptores> ::= <identificador>
<natributosdescriptoress> ::= <natributosdescriptores> <natributosdescriptoress> | ε
<alias> ::= <nalias> <naliass>
<nalias> ::= #gene <identificador> #fuente <identificador>
<naliass> ::= <nalias> <naliass> | ε
<identificador> ::= <letra> <caracteres>
<caracteres> ::= <caracter> <caracteres> | ε
<letra> ::= a | b | c | ... | z | A | B | C | ... | Z
<número> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<caracter> ::= <letra> | <número> | _ | | / | (|) | * | $ | % | & | @

De tal forma que suponiendo que la siguiente relación (PROMOTORES) se encuentra en tercera forma normal esta queda expresada gramaticalmente como se muestra a continuación :

Notación de Codd:

PROMOTORES (DIGITO,NUMAPO,FECING,NOMBRE,REGLEG,RFC,*NUMSUC*,DIGITO_1,NUMAPO_1)

En base a la gramática :

#name_rel
PROMOTORES
#keys
DIGITO
NUMAPO
#endkeys
#desc
FECING
NOMBRE
REGLEG
RFC
NUMSUC
DIGITO_1
NUMAPO_1
#enddesc
#gene
DIGITO_1
#fuente
DIGITO
#gene
NUMAPO_1

#fuente
NUMAPO

7.1.1. Anotaciones sobre la implemetación del proceso de Normalización en lenguaje C

La implementación del módulo de normalización en lenguaje C se publicó previamente en [HERN90] aquí solamente destacaremos los aspectos de interés. La implementación parte de las siguientes estructuras de datos :

```
typedef struct predicate_ {
        char      REL[15];
        char      NEWREL[15];
        struct predicate_   *next;
}
```

La estructura **predicate_** describe el nombre de la relación y sirve para controlar una lista de nuevas relaciones sintetizadas. se inicializan los siguientes apuntadores que son apuntadores a una lista de relaciones descompuestas, una lista de relaciones en tercera forma normal y una lista de relaciones descompuestas en tercera forma normal :

```
predicateptr decomp = NULL;
predicateptr in3nf = NULL;
predicateptr tnfdecomp = NULL;
```

```
typedef struct attr_ {
                    char attr_name[15];
                    struct attr_ *next;
                } attr,*attrptr;
```

La estructura **attr_** contiene le nombre del atributo y sirve para controlar una lista de nombres de atributos

```
typedef struct      fd_ {
                        char      rel_name[15];
                        attrptr   left_side;
                        attrptr   right_side;
                        struct    fd_ *next;
                    } fd, *fdptr;
```

La estructura **fd_** contiene el nombre de la relación a la que la dependencia funcional se aplica , además contiene una lista de los atributos del lado derecho e izquierdo de la dependencia. El algoritmo usado en este trabajo requiere que el lado derecho de las dependencias sean de un solo atributo.

```
typedef struct schema_ {
                    char schema_name[15];
                    attrptr attr;
                    struct schema_ *nextschema;
                } schema, *schemaptr;
```

La estructura **schema_** contiene el nombre de la relación y una lista de nombres de atributos que forman la relación. Además sirve para controlar una lista de esquemas de relación.

```
typedef struct listalhs_ {
```

```
    attrptr leftside;
    struct listalhs_ *next;
   } listalhs, *listptr;
```

La estructura **listalhs_** contiene un apuntador a una lista de atributos que forman el lado izquierdo de la dependencia funcional y sirve para controlar una lista de los lados izquierdos de todas las dependencias funcionales.

```
typedef struct group_ {
                    char      group_name[15];
                    listptr lista;
                    attrptr    at;
                    struct  group_       *nextgroup;
                } group,*ptrgroup;
```

La estructura **group_** contiene el nombre del grupo, una lista de los nombres de atributos que caracterizan al grupo y un apuntador para controlar los lados izquierdos de todas las dependencias funcionales. Además sirve para controlar una lista de grupos.

```
typedef struct closure_ {
                    char clo_name[15];
                    attrptr left_side,clo;
                    struct closure_ *nextclo;
                   } clos, *ptrclosure;
```

La estructura **closure_** contiene el nombre de la relación, un apuntadora una lista de atributos del lado izquierdo de una dependencia funcional y un apuntador a una lista de atributos que forman la cerradura de dicho lado izquierdo. Además sirve para controlar una lista de las cerraduras de todos los lados izquierdos.

```
typedef struct key_ {
                    char key_name[15];
                    attrptr attr_key;
                    struct key_ *nextkey;
                } key, *keyptr;
```

La estructura **key_** contiene el nombre de la relación y una lista de los nombres de los atributos que constituyen la llave para esa relación, además sirve para controlar una lista de las llaves para todas las relaciones sintetizadas.

A continuación presentamos la implementación de los algoritmos principales :

Algoritmo CERRDURA

```
attrptr CERRADURA(char *REL, attrptr X)
{
 fdptr    p = df;                          /*df: conjunto de dependencias de entrada *7
 attrptr W = attr_copy(X);          7* W: conjunto de atributos que forman la Cerradura */
 while(p){
  if(strcmp(p->rel_name,REL) == 0)
    if(subset(p->left_side,W) && !subset(p->right_side,W))
W = setunion(W,p->right_side);
   p = p->next;
 }
 return(W);
```

```
}
```

Inicialmente X se asigna a la cerradura W. Para cada dependencia funcional se evalúa si su lado izquierdo es un subconjunto de W y si su lado derecho no lo es; si ocurre lo anterior el lado derecho se une a la cerradura.

Algoritmo EXTRAÑOS (parte 1)

```
void elimattr(char *REL)
{
 fdptr p;
 attrptr NEWLHS = NULL,Z;
 p=fdcopy(df);                  /*p=df conjunto de dependencias */
 while(p){                      /*para todas las dependencias */
    NEWLHS = reducelhs(REL,p->left_side,p->right_side);
    if(!equal(p->left_side,NEWLHS)){
df = minus1(df,p);  /*se elimina la dependencia */
p->left_side = attr_copy(NEWLHS);
df = asserta(df,p);
    }
    p = p->next;
 }
}
```

La función **void elimattr(char *REL)** toma el conjunto de dependencias funcionales (df) y para cada dependencia funcional llama a la función **attrptr reducelhs(char *REL, attrptr LHS, attrptr RHS)** para eliminarlos atributos extraños del lado izquierdo de la dependencia. Si encuentra atributos extraños, elimina la dependencia funcional del conjunto de entrada y anexa una nueva dependencia al conjunto de dependencias (df), que contiene un nuevo lado izquierdo sin atributos extraños.

Algoritmo EXTRAÑOS (parte 2)

```
attrptr reducelhs(char *REL, attrptr LHS, attrptr RHS)
{
 attrptr A=LHS,Z,ZCLO,Z1,NEWLHS;
 NEWLHS=attr_copy(LHS);
 Z1=attr_copy(LHS);
 while(A){
  Z = minus1_(Z1,A);
  if(Z){
   ZCLO = CERRADURA(REL,Z);
   if(subset(RHS,ZCLO)) NEWLHS = reducelhs(REL,Z,RHS);
  }
  A=A->next;
 }
 return(NEWLHS);
}
```

El algoritmo busca un atributo A en el lado izquierdo de la dependencia funcional y construye la diferencia (Z) entre el lado izquierdo y el atributo y evalúa la cerradura de dicha diferencia (ZCLO) prueba si el lado derecho es un subconjunto de esta cerradura, y si esto sucede el atributo A es extraño, y el algoritmo se llama recursivamente hasta que la reducción no pueda terminar.

Algoritmo ELIMINA

```
void elimredundfds(char *REL)
{
 fdptr p,p0,p1;
 attrptr z,LHS;
 p = fdcopy(df);     /*conjunto de dependencias de entrada *7

 while(p){           /*para todas las dependencias de entrada *7
  LHS=attr_copy(p->left_side);
  df=minus1(df,p);
  z=CERRADURA(REL,LHS);
  if(!subset(p->right_side,z))
       df=asserta(df,p);
  p=p->next;
 }
}
```

La función toma cada una de las dependencias funcionales y la elimina del conjunto de entrada; calculando consecuentemente la cerradura del lado izquierdo de la dependencia respecto a este conjunto reducido. Si el lado derecho está contenido en esta cerradura entonces la dependencia es redundante y se elimina.

7.2. DISEÑO FISICO DE LA BASE DE DATOS

El diseño físico de la base de datos es un proceso que guarda una estrecha relación con el Manejador de Base de Datos donde se vaya a implementar el diseño lógico generado. En el diseño original del SDBD se pensó generar el diseño físico de la base de datos a través de una herramienta conocida como Descriptor de Archivos [CHAP85] debido a que esta era la forma como LIDA (Lenguaje Iconográfico para Desarrollo de Aplicaciones) [CHAP91] describía sus bases de datos y SDBD es una herramienta que se incorporará como utilería para LIDA. Sin embargo, debido a que en modificaciones posteriores de LIDA se desarrollo en CINVESTAV un Manejador de Bases de Datos Relacionales(MBDR) propietario se decidió generar el diseño físico para el nuevo MBDR y ajustarse a la forma en que este último describe las bases de datos físicas.

La definición de una base de datos física para el anterior MBDR consta de 3 componentes principales :

√ La palabra clave **#esquema** seguida de un identificador del nombre de la base de datos que se está realizando el diseño físico.

√ La definición de la base de datos física propiamente dicha, es decir, el cuerpo del diseño físico.

√ La palabra clave **#fin esquema** seguida de un identificador del nombre de la base de datos que se está realizando el diseño físico. Esta palabra clave junto con el primer componente sirven exclusivamente para delimitar el cuerpo del diseño físico.

El cuerpo del diseño físico donde se detallan todas las especificaciones del diseño físico consta de cuatro apartados principales : Diccionario , Archivos, Llaves y Alias, los cuales se describen a continuación :

√ El componente **DICCIONARIO** tiene exactamente la misma definición que la dada para el componente **DA** del **SDBD** en el apartado **2.**

√ En el componente **ARCHIVOS** se definen los campos que integraran los archivos de la nueva base de datos.

Este componente puede constar de la definición de uno o más archivos, donde un archivo se define con la palabra clave **#archivo** seguida de uno o más campos identificadores, donde el primer campo identificador denota el nombre del archivo y finalizado con la palabra clave **#fin archivo**

√ El tercer componente **LLAVES** puede contar de la definición de una o más llaves, donde una llave se define con la palabra clave **#llave** seguida de un identificador que denota el nombre del archivo y de uno o más identificadores de campos que denotan las llaves del archivo.

√ El último componente **ALIAS** puede contar de la definición de uno o más alias de nombres de atributos, donde un alias se define por la palabra clave **#alias** seguido de dos identificadores de campos donde el primer identificador denota el campo generado (campo alias) a partir de un campo fuente (segundo identificador)

Para la definición de este lenguaje para diseño físico de base de datos se desarrollo una gramática, la cual se presenta a continuación :

<Base de datos> ::= #esquema <identificador> <definicion> #fin esquema <identificador>
<definicion> ::= <diccionario> <archivos> <llaves> <alias>
<diccionario> ::= #dicionario <atributos> #fin diccionario
<atributos> ::= <sentencia><sentencias>
<sentencia> ::= <identificador> . <tipo> , <longitud>, <máscara>
<sentencias> ::= <sentencia> <sentencias> | ε
<longitud> ::= <número> <longitudes>
<longitudes> ::= <número> <longitudes> | ε
<máscara> ::= " <caracteres> "
<identificador> ::= <letra> <caracteres>
<caracteres> ::= <caracter> <caracteres> | ε
<tipo> ::= A | D | N | Z | C
<letra> ::= a | b | c | ... | z | A | B | C | ... | Z
<número> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<caracter> ::= <letra> | <número> | _ | . | / | (|) | * | $ | % | & | @
<archivos> ::= <narchivo> <narchivos>
<narchivo> ::= #archivo <campos> #fin archivo
<narchivos> ::= <narchivo> <narchivos> | ε
<campos> ::= <ncampo> <ncampos>
<ncampo> ::= <identificador>
<ncampos> ::= <ncampo> <ncampos> | ε
<llaves> ::= <nllave> <nllaves>
<nllave> ::= #llave <identificador> <atributosllave>
<nllaves> ::= <nllave> <nllaves> | ε
<atributosllave> ::= <natributosllave> <natributosllaves>
<natributosllave> ::= <identificador> [, <identificador]*
<natributosllaves> ::= <natributosllave> <natributosllaves> | ε
<alias> ::= <nalias> <naliass>
<nalias> ::= #alias <identificador> <identificador>
<naliass> ::= <nalias> <naliass> | ε

En base a la anterior gramática una base de datos típica podría ser expresada de la siguiente forma:

```
#esquema SAPF
#diccionario
        NOMBRE,   A,  30,     "_____"
        DIRECC,   A,  30,     "_____"
```

```
        CODPOS,   Z,   5,      "_____"
        RFC,      A,  14,      "______________"
        NUMCLI,   Z,   7,      "_______"
        FIDUCI,   A,  40,      "________________________________________"
        REPRES,   A,  30,      "______________________________"
        NUMSUC,   Z,   5,      "_____"
#fin diccionario

#archivo FONDOS
        NUMCLI
        FIDUCI
        REPRES
#fin archivo

#archivo SUCURSALES
        NUMSUC
        NOMBRE
        DIRECC
        CODPOS
        RFC
        NUMSUC_1
#fin archivo

#llave  FONDOS      NUMCLI
#llave  SUCURSALES   NUMSUC

#alias  NUMSUC_1     NUMSUC

#fin esquema SAPF
```

7.2.1. Anotaciones sobre la implementación del Diseño Físico de Base de Datos en lenguaje C

El programa de generación del esquema físico trabaja de la siguiente manera: Existen 3 rutinas principales. La primera de ellas lee el diccionario de datos y lo formatea, el resultado de este formateo es escrito en un archivo temporal (tmp13130.rhs, apuntador sal6)

La segunda lee el archivo (.rel) generado por el módulo de normalización y va generando dos archivos temporales. La información que contienen es :

Tmp13139.rhs apuntador sal1 contiene nombres de archivos y sus campos
Tmp13138.rhs apuntador sal4 contiene las llaves de los archivos y los alias de los campos

La tercera toma la información de los archivos temporales anteriores y genera el archivo resultado (.dbd). controlado por el apuntador sal5.

Las estructuras de datos utilizadas son las siguientes :

```
typedef struct attr_ {
                        char attr_name[15];
                        struct attr_ *next;
                } attr,*attrptr;

typedef struct schema_ {
                        char schema_name[15];
```

```
                    attrptr attr;
                    struct schema_ *nextschema;
                } schema, *schemaptr;

/*  Archivos de entrada y salida. */
FILE *ent1,*ent2,*sal1,*sal2,*sal4,*sal5,*sal6;

/* Apuntador a la lista de esquemas */
schemaptr schemas = NULL;

/*  Apuntadores a listas de atributos auxiliares durante la lectura
    de informacion de los archivos : *.rel. */
attrptr l1,l2,l3;
```

Donde el tipo **attrptr** es un apuntador a una lista de estructura de atributos y una estructura de atributo se define con una nombre de 15 caracteres y un apuntador al siguiente nombre y el tipo **schemaprt** es un apuntador a una lista de estructuras de esquemas (relaciones) que se define como un nombre de 15 caracteres un apuntador a una lista de atributos y una apuntador al siguiente esquema.

La función principal del programa es la siguiente :

```
main()
{
  char *sincar;
  int longi,i;
  textbackground(BLUE);
  clrscr();
  textcolor(WHITE);
  sincar=(char *)malloc(40);
  strcpy(sincar,"");
  cad=(char *)malloc(10);
  strcpy(cad,"");
  caddic=(char *)malloc(10);
  strcpy(caddic,"");
  cad1=(char *)malloc(10);
  strcpy(cad1,"");
  cad2=(char *)malloc(10);
  strcpy(cad2,"");
  entrada=(char *)malloc(40);
  strcpy(entrada,"");
  caddic=get_namedic();
  cad=get_name();
  if(strcmp(cad,"")){
    strcpy(cad1,cad);
    strcpy(cad2,cad);
    strcpy(cad3,caddic);
    strcat(cad3,".dic");
    strcat(cad1,".3fn");
    ent1=fopen(cad1,"r");
    ent2=fopen(cad3,"r");
    if(ent1 && ent2){
      strcat(cad2,".dbd");
      sal5=fopen(cad2,"w");
```

```
    sal4=fopen("tmp13138.rhs","w");
    sal1=fopen("tmp13139.rhs","w");
    sal6=fopen("tmp13130.rhs","w");
    gen_dics();
    while((schemas = lect_datos())){
          print_res(schemas->schema_name,l1,l3);
          schemaerase(schemas);
          schemas=(schemaptr)0;
          attrerase(l1);
          attrerase(l2);
          attrerase(l3);
          l1=(attrptr)0;
          l2=(attrptr)0;
          l3=(attrptr)0;
    }
    fprintf(sal4,"\n");
    while(entrada){
          if(!strcmp(entrada,"#gene\n")){
            fprintf(sal4,"#alias  ");
            fgets(entrada,40,ent1);
            longi=strlen(entrada);
            strcpy(sincar,entrada);
            sincar[longi-1]='\0';
            fprintf(sal4,"%s",sincar);
            for(i=0;i<=15-longi;i++)
              fprintf(sal4," ");
            fgets(entrada,40,ent1);
            fgets(entrada,40,ent1);
            fprintf(sal4,"%s",entrada);
            fgets(entrada,40,ent1);
          }else
            break;
    }
    fclose(sal1);
    fclose(sal4);
    fclose(sal6);
    fclose(ent1);
    print_resfin();
   }
  }
  free(&cad);
  free(&cad1);
  free(&cad2);
  free(&entrada);
  system("del *.rhs");
  clrscr();
}
```

La función principal funciona de la siguiente manera :

- Obtiene el nombre del diccionario de atributos por utilizar a través de la función get_namedic()
- Obtiene el nombre del archivo en tercera forma normal que sirve de entrada para el diseño físico a través de la función get_name()

- Forma el nombre del archivo definitivo de salida concatenando al archivo de entrada la terminación DBD y lo abre para escritura con el apuntador sal5.

- Abre para salida los archivos temporales tmp13138.rhs, tmp13139.rhs, tmp13130.rhs donde almacenará el diccionario de atributos, la definición de archivos y sus campos respectivos, la definición llaves por archivo y alias de atributos.

- Ejecuta la función gen_dics() la cual tomando como base el diccionario de atributos de entrada lo formatea y lo deja en el archivo temporal tmp13130.rhs

- A través del siguiente while :

```
while((schemas = lect_datos())){
        print_res(schemas->schema_name,l1,l3);
        schemaerase(schemas);
        schemas=(schemaptr)0;
        attrerase(l1);
        attrerase(l2);
        attrerase(l3);
        l1=(attrptr)0;
        l2=(attrptr)0;
        l3=(attrptr)0;
....}
```

Lee un esquema (relación en tercera forma normal) del archivo de entrada cuya terminación es *.rel a través de la función lect_datos(), la cual retorna un apuntador del tipo schemaptr por cada relación leída. Posteriormente formatea la salida e imprime los resultados en los archivos tmp13138.rhs y tmp13139.rhs a través de la función print_res(schemas->schema_name, l1, l3), libera la memoria de los apuntadores al esquema y la lista de atributos e inicializa los apuntadores a nulo, y repite el proceso hasta que no encuentra más información en el archivo de entrada.

- Mientras en el archivo de entrada (*.rel) encuentra la palabra #gene realiza el proceso de formateo de alias de atributos y lo almacena en el archivo temporal tmp13138.rhs

- Por último llama a la función print_resfin() la cual genera el archivo de diseño físico de base de datos con terminación *.dbd utilizando como entrada los archivos temporales generados.

Cabe señalar que el proceso de diseño físico al ser una actividad totalmente dependiente del MBDR que se este utilizando, no está limitado a realizarse para un sólo MBDR, si que es posible generar el diseño físico de la base de datos para cuantos MDBR que se desee siempre y cuando se conozca a detalle la gramática conque están construidos. En nuestro caso desarrollamos el diseño para la gramática propuesta pero es factible desarrollar los módulos de diseño físico que se deseen.

8. DESCRIPCION DEL SISTEMA PARA DISEÑO DE BASES DE DATOS EVE

El sistema para diseño de base de datos EVE fue desarrollado en turbo C y se implemento una versión para Windows 3.1 donde se creó un grupo llamado EVE que agrupa a cinco iconos que permiten el acceso a las funciones básicas del sistema, tal como se muestra en la figura 3.

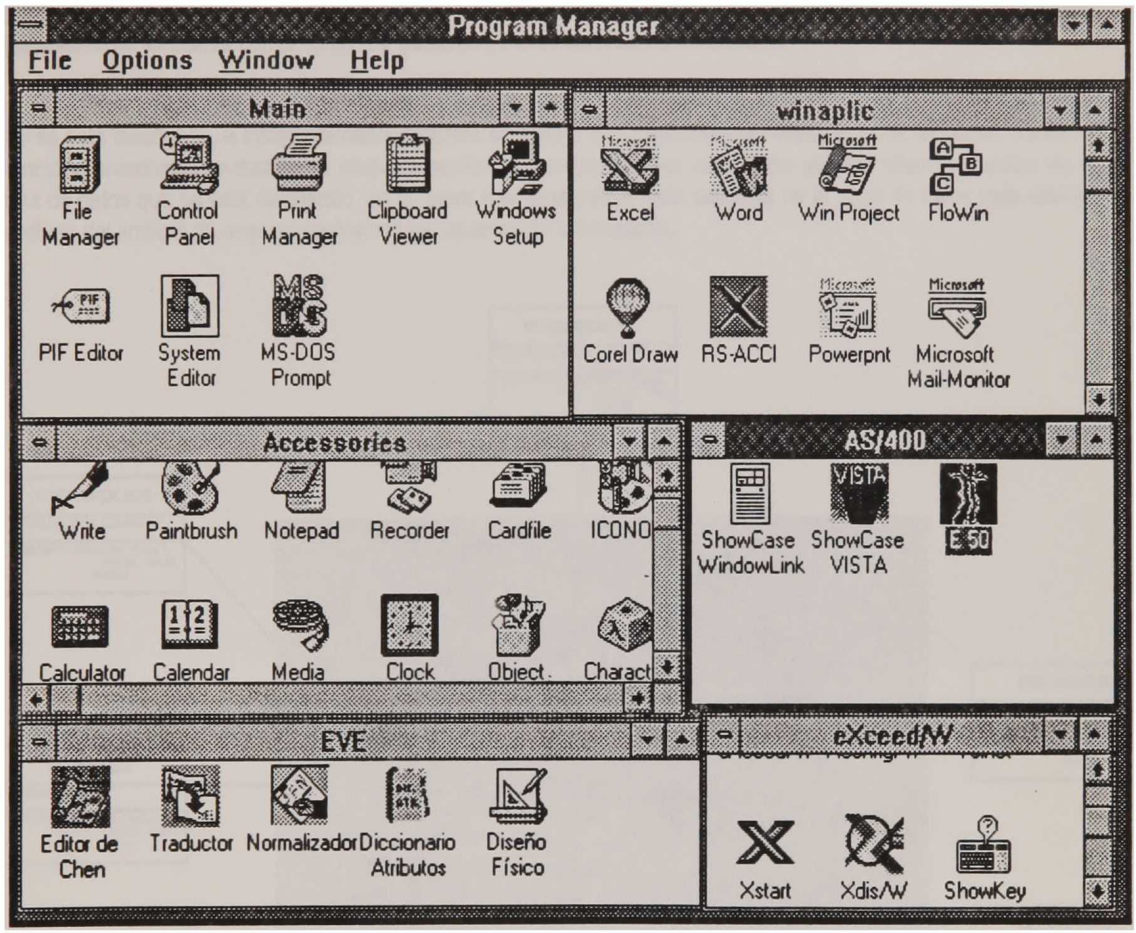


Figura 3 : Grupo EVE en Windows 3.1 para el SDBD-EVE

- **Editor de Chen**: Accesa el editor de diagramas de Chen y valida que los diagramas de base de datos editados cumplan las reglas de construcción de Chen y que todos los atributos de las entidades pertenezcan a un diccionario de atributos.

- **Traductor** : Realiza la lectura de un diagrama de Chen de una base de datos y su transformación en un esquema de relaciones de Codd sin normalizar.

- **Normalizador**: Realiza la normalización de la relaciones de Codd generadas en el proceso de traducción y genera un esquema de relaciones de Codd en tercera forma normal.

- **Diccionario Atributos** : Realiza la edición, compilación y generación de diccionarios de atributos que sirven de entrada para el proceso de edición de diagramas de Chen y el diseño físico de la base de datos.

- **Diseño Físico** : Realiza el proceso de diseño físico de la base de datos para el manejador de base de datos utilizado por LIDA (Versión para workstation).

Para describir el uso de SDBD-EVE se presentará el diseño de una base de datos de una compañía que se encarga de administrar portafolios financieros de inversión de clientes que son operados por diversos grupos financieros. Los clientes pueden ser nacionales, internacionales o fondos, donde fondos se refiere a clientes corporativos. Para ello la compañía cuenta con un grupo de promotores agrupados en varias sucursales, adicionalmente se cuenta con un grupo de gestores para los clientes internacionales.

Para emplear SDBD primeramente se tiene que realizar el análisis de requerimientos de la base de datos que se está diseñando, e inmediatamente después se pasa a la construcción del diccionario de atributos. Como se mencionó anteriormente durante la etapa de análisis de requerimientos se pueden generar diferentes vistas de la base de datos que se está diseñando, en la figura 4 se presenta el vista definitiva de la base de datos bajo diseño, producto del análisis de requerimientos con los usuarios de la compañía.

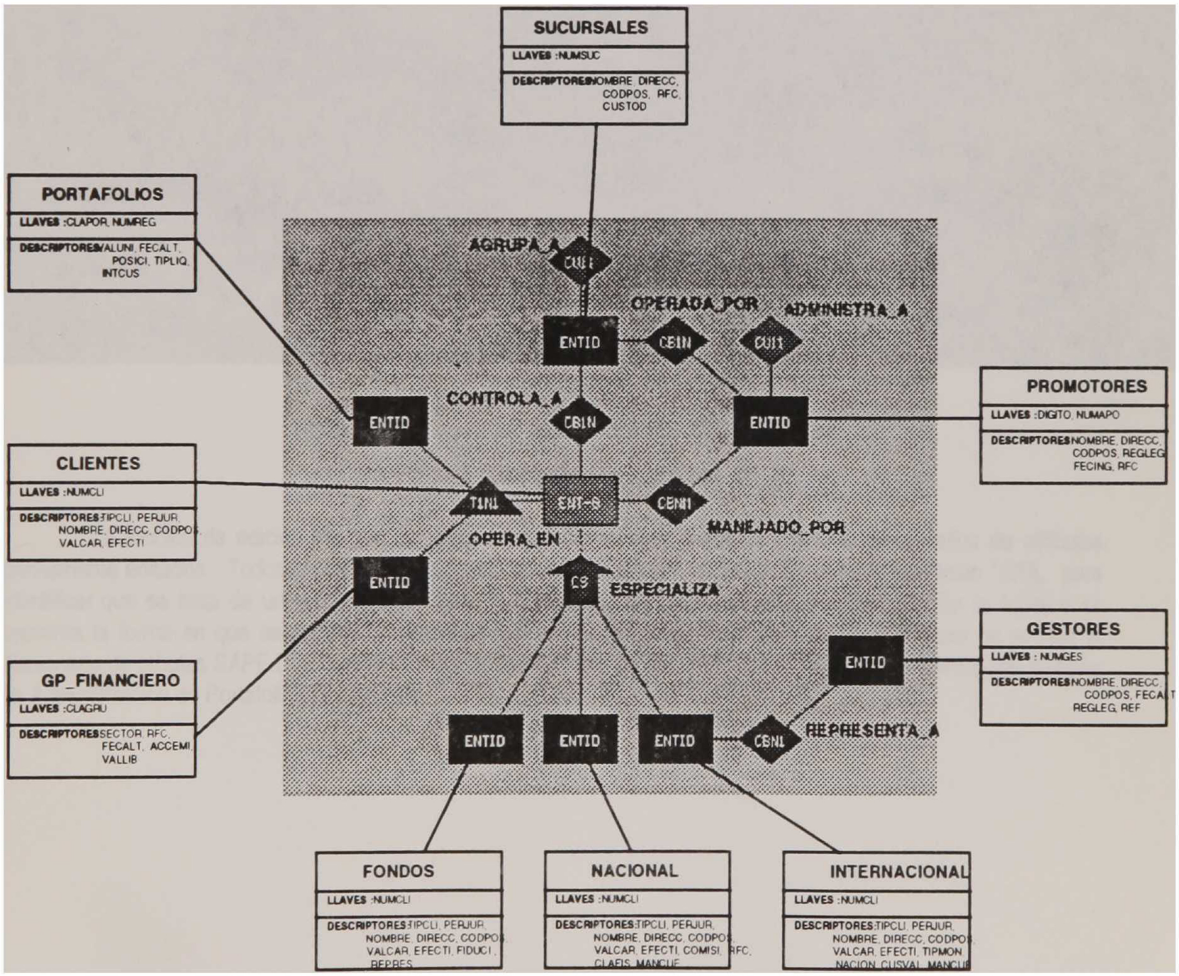


Figura 4: Vista definitiva de la base de datos de Administración de Portafolios Financieros (SAPF)

8.1. DICCIONARIO DE ATRIBUTOS

El SDBD ofrece una herramienta para la edición y manipulación del diccionario de atributos, la cual es accedida a través del icono **Diccionario Atributos.** La herramienta para la manipulación del diccionario de atributos consta básicamente de un editor de texto y un compilador que determina si el diccionario cumple las reglas gramaticales. En la figura 5 se presenta el menú principal del Diccionario de Atributos y la opciones de edición.

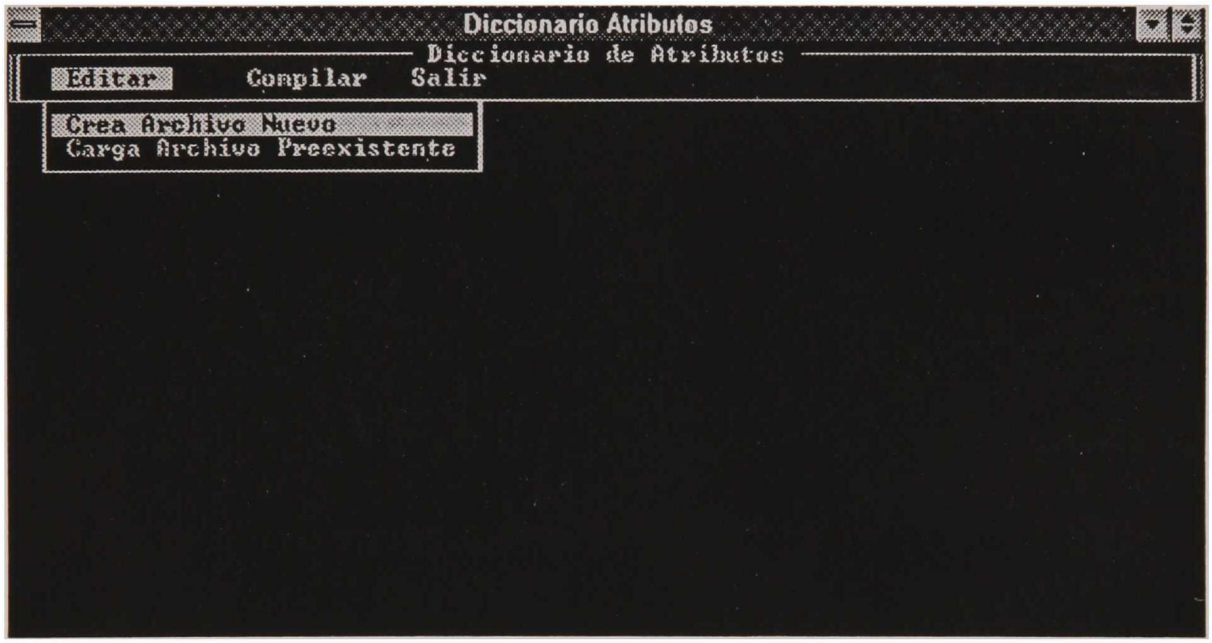


Figura 5 : Diccionario de Atributos

La función de edición me permite crear diccionarios nuevos o abrir y modificar diccionarios de atributos previamente editados. Todos los diccionarios de atributos editados son almacenados con la extensión *.DDL para identificar que se trata de un archivo fuente para la generación de un diccionario de atributos. En la figura 6 se presenta la forma en que se accesa un diccionario de atributos previamente editado, en este caso se accesa el diccionario de tributos SAPF.DDL que es diccionario de atributos producto del análisis de requerimientos del Sistema de Administración de Portafolios Financieros (SAPF).

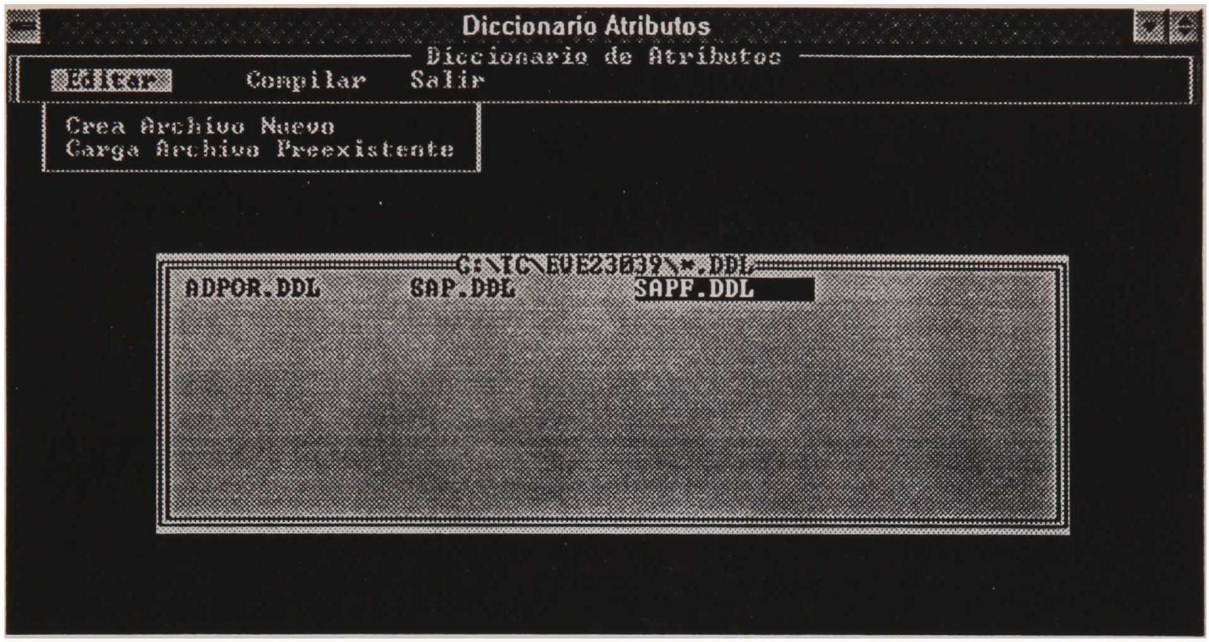


Figura 6 : Acceso a un diccionario de atributos previamente editado

Este diccionario de atributos contiene la descripción de todos los atributos detectados durante el análisis de requerimientos del SAPF. En el se muestra el nombre, el tipo, la longitud y la máscara de captura de todos loa atributos que serán utilizados en el diseño de la base de datos. En la figura 7 se presenta la ventana de edición del diccionario.

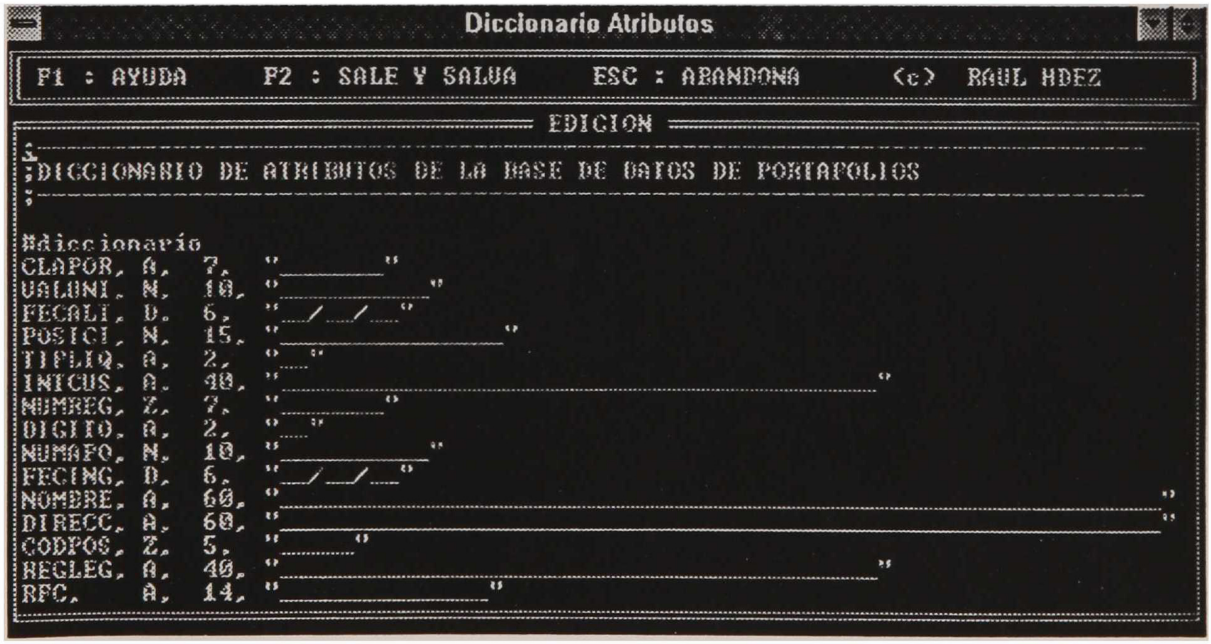


Figura 7: Diccionario de atributos del Sistema de Administración de Portafolios Financieros

Para la creación del diccionario el editor facilita el acceso a la funciones de un editor de texto estandar con funciones de marcado y copiado de bloques y formateo de texto, adicionalmente debido a que el compilador del diccionario acepta el ";" como un comentario se puede detallar en el fuente del diccionario la documentación necesario respecto al significado de los atributos, para mayor detalle en el **Anexo 1** se presenta el fuente completo del diccionario de atributos del SAPF.

Una vez que el diccionario de atributos se ha introducido en el editor, se salva y se procede a la fase de compilación, para ello es necesario seleccionar el diccionario de atributos que se desea compilar tal como se muestra en la figura 8.

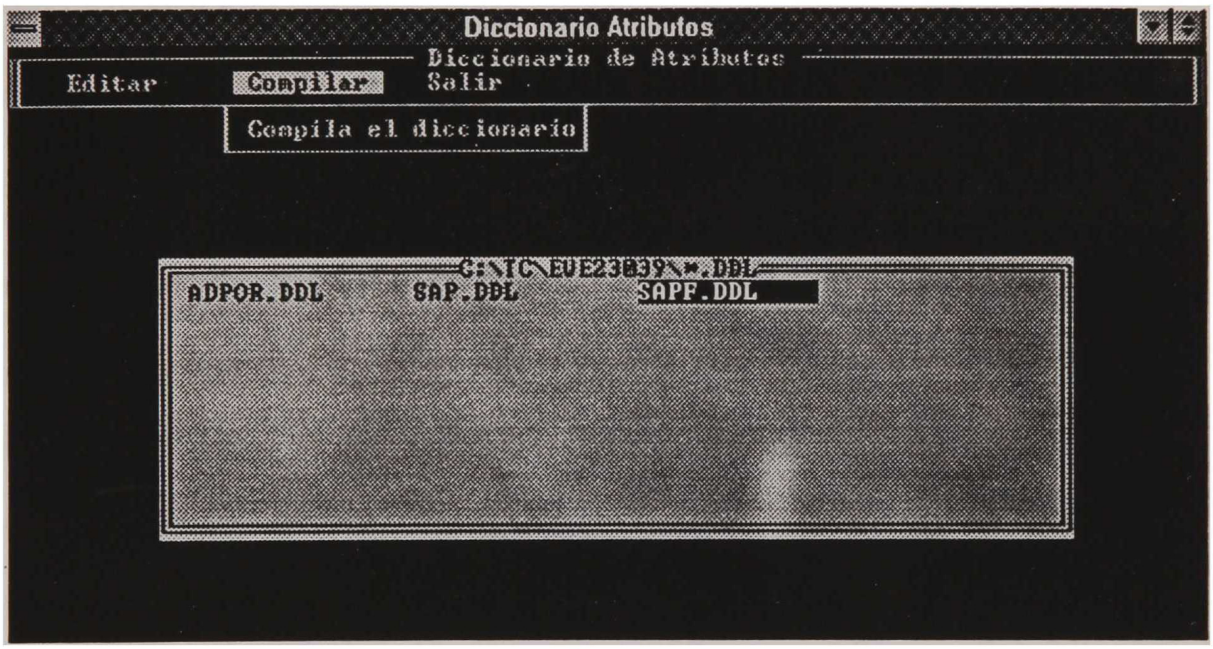


Figura 8: Compilación de diccionario de atributos del SAPF

En caso de que se presentara algún problema durante esta fase el compilador presenta una lista de los errores cometidos durante la edición señalando la línea y el tipo de error que se trata. Para corregir dichos errores será necesario editar nuevamente el diccionario y ejecutar la compilación hasta que quede libre de errores. Una vez que el diccionario de atributos es compilado con éxito se genera un diccionario de atributos objeto con extensión *.DIC el cual es utilizado por editor de Chen y el proceso de diseño físico de la base de datos. En el **Anexo 2** se presenta el diccionario de atributos objeto generado para el SAPF.

8.2. EDITOR DE CHEN

Posterior a la creación del diccionario de atributos se procede a la creación del diagrama de Chen para el Sistema de Administración de Portafolios Financieros (SAPF). Previamente con los usuarios se habrán discutido diferentes vistas del modelo de datos del SAPF hasta que se procede a la edición de la vista definitiva tal como se muestra en la figura 4. El editor de Chen es accesado a través de icono **Editor de Chen** con esta herramienta del SDBD se pueden editar y analizar todos los modelos y vistas de base de datos que sean necesarios, en este caso mostramos el uso básico del editor de Chen con la edición de la vista definitiva del modelo de datos del SAPF. El editor de Chen ofrece 3 funciones básicas : Archivos, Edición y Análisis, La función de archivos del menú principal permite cargar archivos previamente editados o salvar un archivo editado recientemente. El sistema almacena todos los archivos editados con la extensión *.CHN, lo cual los identifica como fuentes de un diagrama de Chen. En la figura 9 presentamos el menú principal del editor de Chen y la manera como es cargado el diagrama de Chen denominado SAPF. Cabe señalar que este editor permite cargar un archivo previamente editado, realizar modificaciones sobre él y salvarlo con otro nombre; de tal forma que diagramas previamente editados pueden servir de base para la edición de otros diagramas.

El Editor de Chen consta de tres área básicas: El área de Menúes, el área de Edición y el área de Selección de Iconos y Atributos. El área de menúes que es la que se presenta en la figura 9 permite elegir cualquier opción del menú para realizar alguna función en especifico. El área de Edición es el espacio físico del editor donde se realiza la construcción de los diagramas de Chen y por último el área de selección de iconos y atributos es una parte de editor que permite seleccionar una entidad o conector que formará parte de la vista definitiva del diagrama que se este editando, además de que es el área donde se asignan nombres y atributos a las entidades y conectores y por último tiene la función de presentar los mensajes de error producto del análisis sintáctico de un diagrama de Chen. Por comodidad las llamaremos Area I,II y III respectivamente.

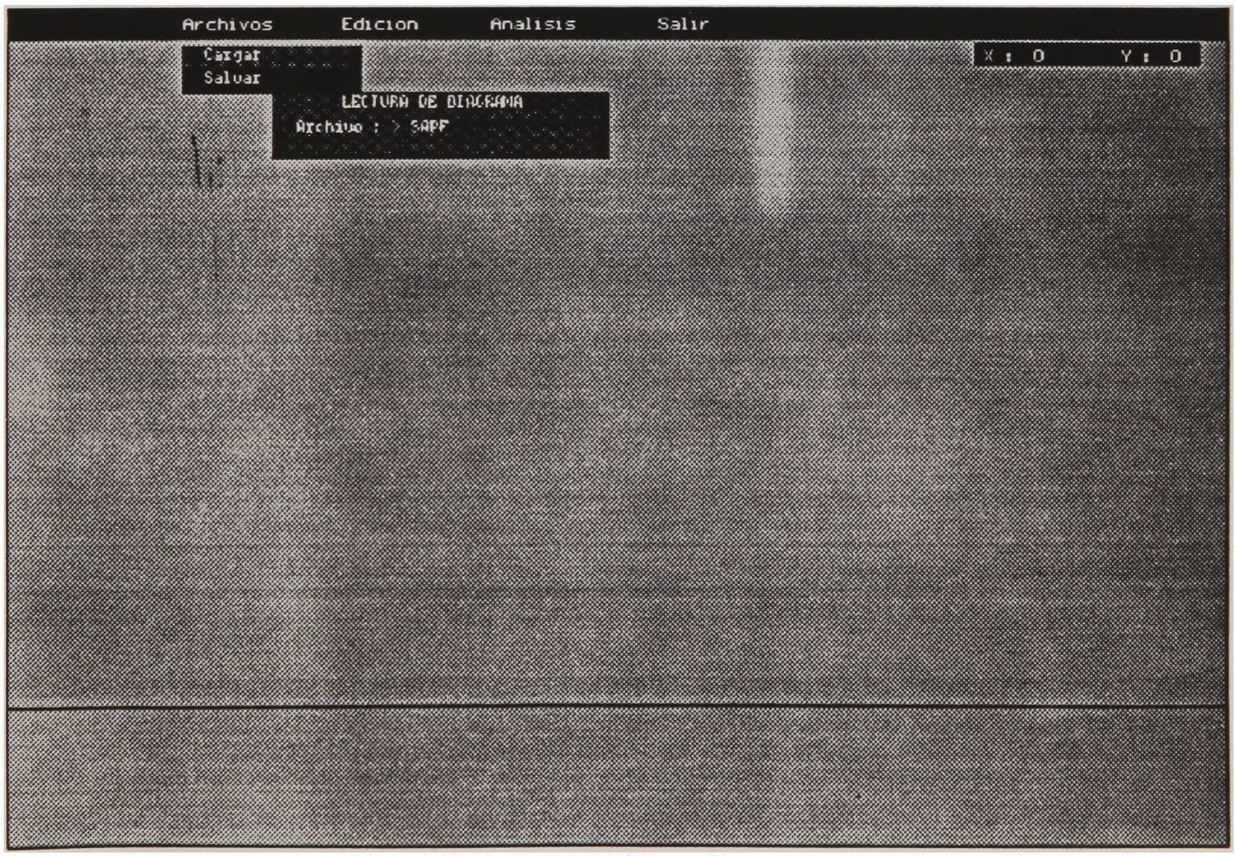


Figura 9: Menú principal del Editor de Chen.

Para la edición de diagramas el Editor de Chen ofrece varias herramientas que se agrupan en el menú de Edición, las cuales se describen a continuación :

- **Escala:** Esta función permite seleccionar la escala del dibujo que se está editando, pudiendo ser del 25%, 50% o 100% del tamaño normal del diagrama editado, lo anterior permite tener una vista completa del diagrama en el caso de que este sea demasiado grande. Opcionalmente el editor tiene una función de scroll que permite ir visualizando partes del diagrama moviéndose a través del él con el uso de las flechas.

- **Insertar <Ctrl I>:** Seleccionando la opción de insertar o presionando las teclas de <Ctrl> e <I> simultáneamente se activa la opción de inserción la cual activa en el sector de abajo del área de edición un menú de iconos por seleccionar. Para seleccionar un icono es necesario moverse con las flechas y una vez posicionados en el icono deseado seleccionar este presionando la tecla de <ENTER>. Si selecciona un conector unario <CU> ó un conector binario <CB> ó un conector ternario <CT> se presentará otro submenú que contiene los diferentes conectores posibles de seleccionar ya sea <CU11>, <CU1M>, <CUMN>, ..., <CTMNM>.

- **Borrar <Ctrl B>:** Seleccionando la opción de borrar o presionado las teclas de <Ctrl> y <B> simultáneamente se elige la opción de borrar. Para que esta opción funcione el cursor deberá estar ubicado en alguna ENTIDAD y si esta condición se cumple se elimina del diagrama editado la ENTIDAD y todas sus relaciones (conexiones) con otras ENTIDADES.

- **Nombre <Ctrl N>:** Seleccionando esta opción o presionando las teclas de <Ctrl> y <N> simultáneamente se puede asignar un nombre de identificación a una entidad o a un conector. Para que esta función se active el cursor deberá estar ubicado sobre una entidad o un conector, con lo cual activará en la ventana de atributos la opción de digitar con caracteres alfabéticos un nombre que será asignado a la ENTIDAD. Si esta opción es seleccionada sobre alguna ENTIDAD o conector que ya tenía asignado un nombre su nombre es modificado con el nuevo nombre que sea introducido.

- **Descriptores <Ctrl D>:** Seleccionando esta opción o presionando las teclas de <Ctrl> y <D> simultáneamente se pueden asignar atributos descriptores a una ENTIDAD. Para que esta opción sea activada será necesario ubicarse sobre alguna ENTIDAD sobre el diagrama con el cursor, si se ubica sobre algún conector esta opción no surtirá efecto. Una vez posicionado sobre alguna entidad esta opción activa el área III para edición de caracteres alfabéticos, cada atributo descriptor es separado con la tecla de <ENTER>. Si se posiciona sobre una entidad que ya tenía asignada una lista de atributos descriptores con esta acción la lista anterior será eliminada y se podrá asignar una nueva lista.

- **Llave <Ctrl L>:** Seleccionando esta opción o presionando las teclas de <Ctrl> y <L> simultáneamente se pueden asignar atributos llaves a una ENTIDAD. Para que esta opción sea activada será necesario ubicarse sobre alguna ENTIDAD sobre el diagrama con el cursor, si se ubica sobre algún conector esta opción no surtirá efecto. Una vez posicionado sobre alguna entidad esta opción activa el área III para edición de caracteres alfabéticos, cada atributo llave es separado con la tecla de <ENTER>. Si se posiciona sobre una entidad que ya tenía asignada una lista de atributos llaves con esta acción la lista anterior será eliminada y se podrá asignar una nueva lista.

- **Conectar <Ctrl C>:** Seleccionando esta opción o presionando las teclas de <Ctrl> y <C> simultáneamente se estará en posibilidad de conectar ENTIDADES de diagrama a través de conectores de relación Unarios, Binarios, Ternarios de Especialización o de Generalización. Para ello será necesario ubicarse con el cursor sobre el conector, ya que esta opción no será activado si el cursor esta ubicado sobre la ENTIDAD, la regla del asunto anterior se debe al hecho que el control de las conexiones las lleva el conector y no la entidad. Una vez ubicado sobre algún conector el cursor quedará fijo sobre el conector y un nuevo cursor aparecerá en la esquina superior izquierda del área de edición el cual nos permitirá seleccionar la ENTIDAD que se desea conectar con

esta RELACIÓN. Cabe señalar que este editor realiza un análisis sintáctico al momento de edición, de tal suerte que si se desea conectar más de dos entidades a un conector binario se marcará error y no se podrá realizar la conexión. Existe un sólo caso que viola la regla anterior y esta en hecho de conectar una ENTIDAD DE GENERALIZACION a un conector de ESPECIALIZACION o GENERALIZACION, ya que la conexión se realiza de la ENTIDAD hacia el CONECTOR y no al revés como se explicó anteriormente, el hecho se debe a que se trata de una conexión especial, que en el diagrama editado se representa con líneas punteadas, que permite controlar el hecho de que no existan anidamientos en las generalizaciones o especializaciones.

- **Desconectar <Ctrl X>:** Seleccionando esta opción o presionando las teclas <Ctrl> y <X> simultáneamente se permite eliminar ligas entre conectores y entidades. Para que esta función se active será necesario ubicarse con el cursor sobre algún conector lo cual ocasionará que el cursor se quede fijo sobre el conector y un nuevo conector aparezca en la esquina superior izquierda del área de edición, el cual no permitirá seleccionar la ENTIDAD que se desea sea desconectada. Cabe señalar que la funcionalidad de esta opción esta incorporada en la opción de Borrar.

- **+ Llave:** Esta opción permite adicionar un atributo llave a una lista de atributos llave de alguna ENTIDAD, para que esta opción sea activada será necesario ubicarse con el cursor sobre alguna ENTIDAD.

- **- Llave:** Esta opción permite eliminar un atributo llave a una lista de atributos llave de alguna ENTIDAD, para que esta opción sea activada será necesario ubicarse con el cursor sobre alguna ENTIDAD.

- **+ Descriptor:** Esta opción permite adicionar un atributo Descriptor a una lista de atributos Descriptores de alguna ENTIDAD, para que esta opción sea activada será necesario ubicarse con el cursor sobre alguna ENTIDAD.

- **- Descriptor:** Esta opción permite eliminar un atributo Descriptor a una lista de atributos Descriptores de alguna ENTIDAD, para que esta opción sea activada será necesario ubicarse con el cursor sobre alguna ENTIDAD.

A continuación describiremos la forma de editar el diagrama de SAPF utilizando las herramientas de edición previamente descritas:

Primeramente se procede a crear la vista entera del diagrama SAPF para ello nos ubicamos con el cursor en algún punto de interés en el área de edición y seleccionando la opción de Inserción se activará el área III del editor y seleccionaremos un icono de ENTIDAD posicionándonos sobre él y presionando la tecla de <ENTER>, repitiendo los pasos anteriores podremos editar todas las entidades y conectores que necesitemos. A continuación usando la opción de Conectar podremos conectar las entidades a los conectores tal como se muestra en la figura 10.

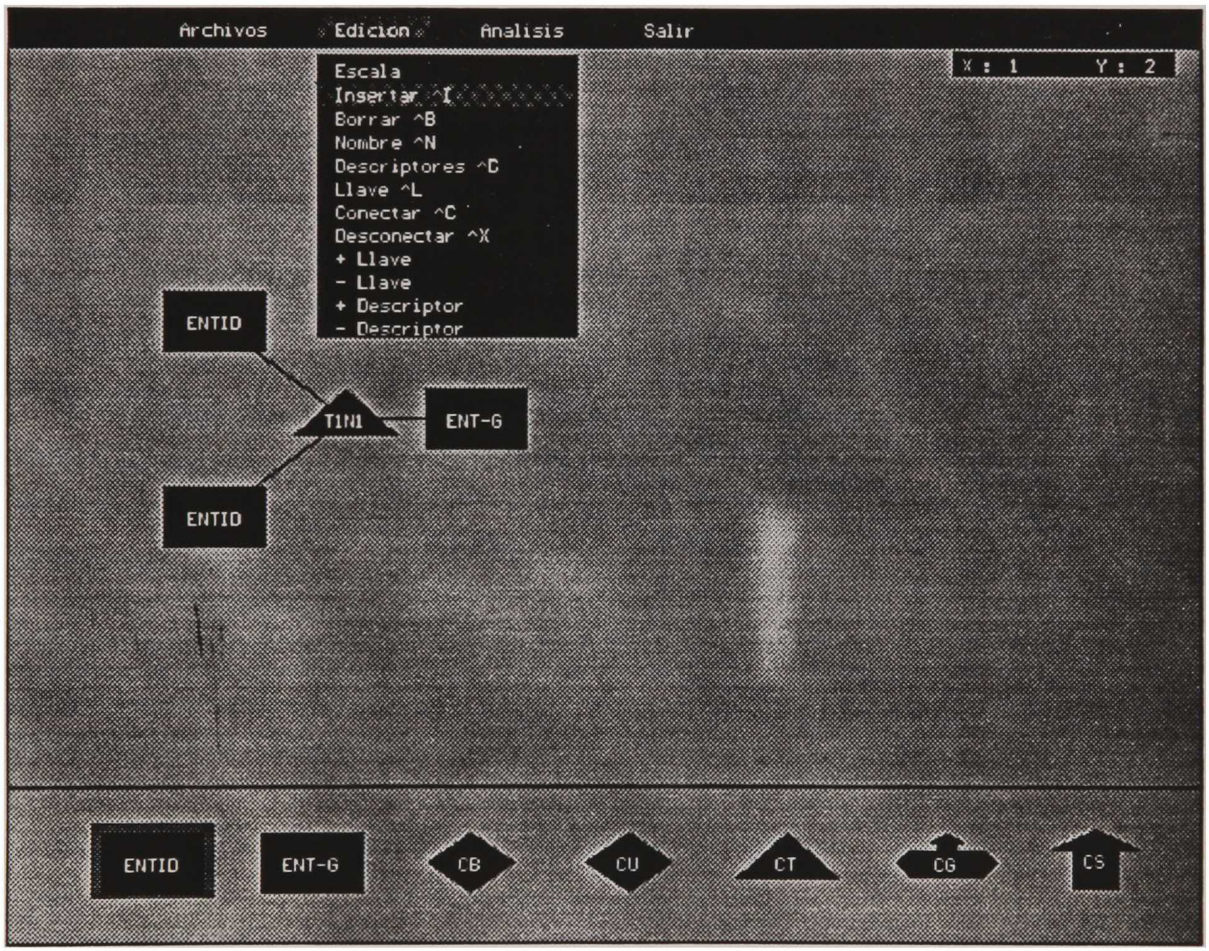


Figura 10: Herramientas del editor de Chen

Posteriormente podremos asignar Nombres a las ENTIDADES y CONECTORES y atributos descriptores y llaves a las ENTIDADES utilizando las opciones de **Nombre, Descriptores y Llave** del menú de edición tal como se muestra en la figura 11.

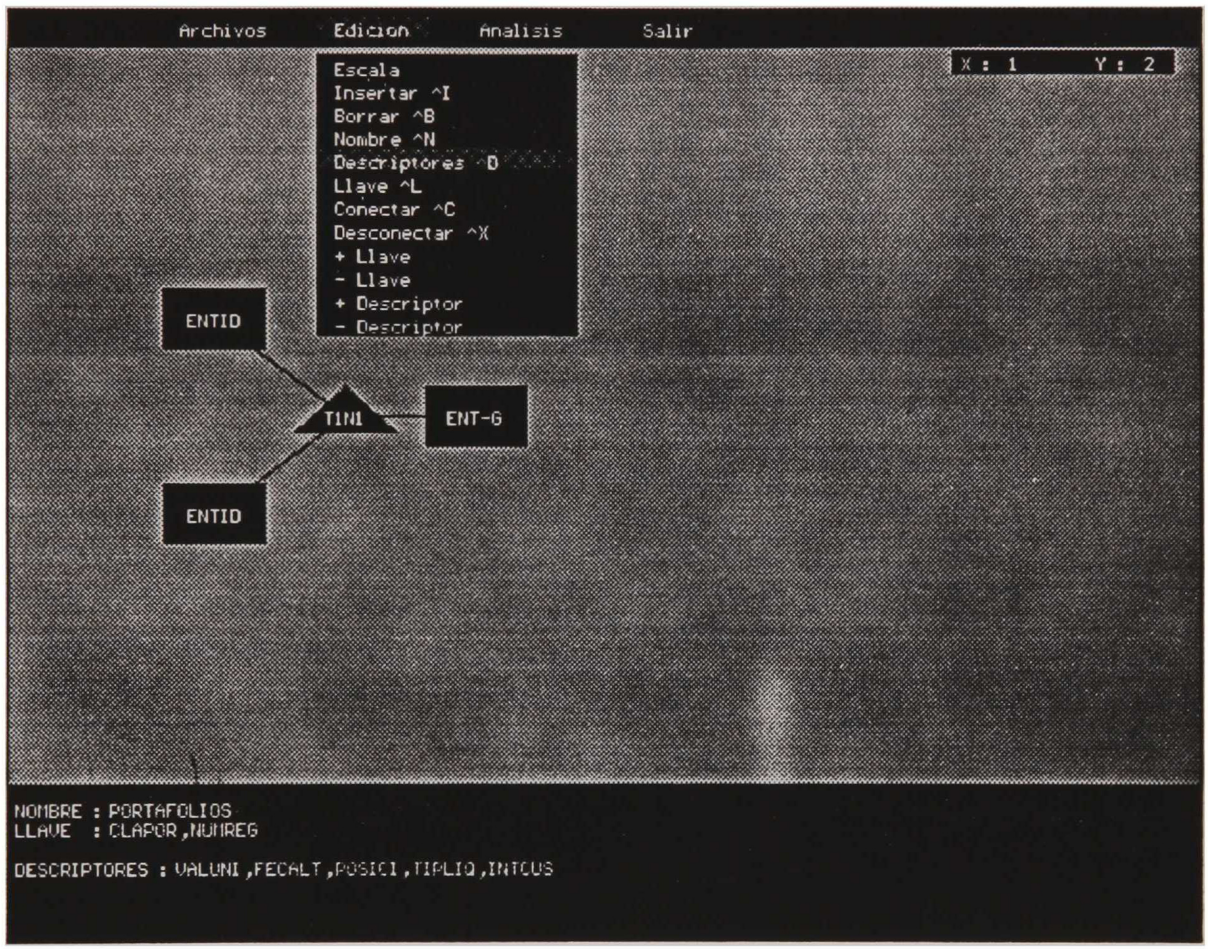


Figura 11: Asignación de nombres a Entidades y Conectores

Una vez realizados los pasos anteriores se habrá creado la vista definitiva del diagrama del SAPF, percátese de que ha editado todas las ENTIDADES y CONECTORES que necesita y las conexiones entre las entidades son las correctas, también revise que todas las entidades y conectores tienen un nombre asignado y que las listas de atributos descriptores y llaves de cada ENTIDAD están completos. Una vez que se cercioró de los aspectos anteriores habrá terminado la edición de una diagrama entidad-relación de Chen, que en nuestro caso se trata del diagrama del SAPF tal como se muestra en la figura 12.

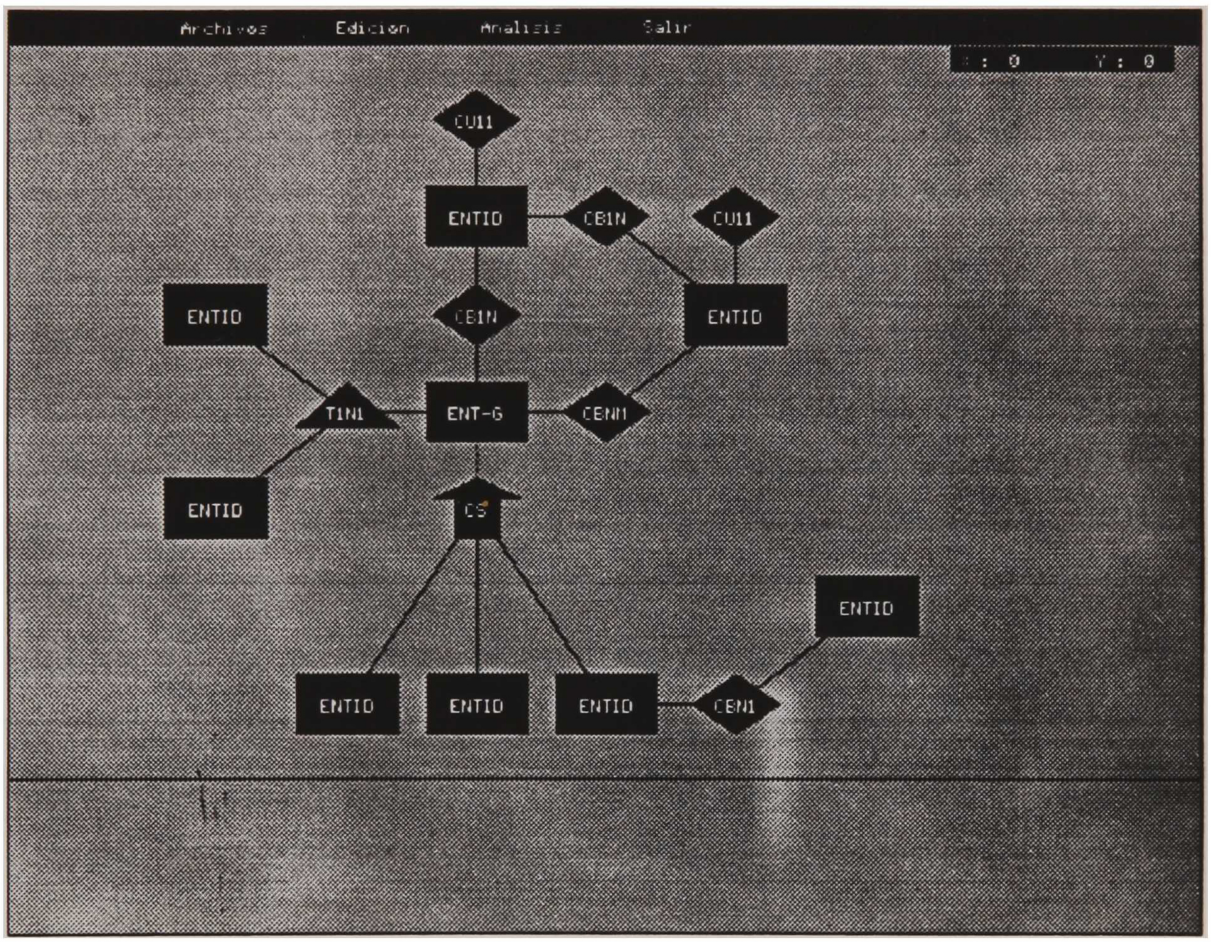


Figura 12: Edición definitiva de Diagrama de Chen

Generalmente por más que cuidemos nuestro trabajo es común cometer errores en la edición de diagramas de Chen, para el área de menúes se proporcionó la opción de ANALISIS la cual realiza un parsing sobre el diagrama verificando que todas las reglas para editar un diagrama de Chen se cumplan, adicionalmente en esta opción se pide indicar el nombre del diccionario de atributos sobre el cual se basan las listas de atributos descriptores y llaves. Si existieran errores en la construcción del diagrama estos serán serán desplegados en el área III de editor indicando el tipo de error que se trata y las coordenadas del diagrama en el cual se esta cometiendo el error, recuerde que si algún atributo no se localiza en el diccionario de atributos esto será indicado como un error en la coordenada del editor donde se localiza la ENTIDAD. La opción de análisis se muestra en la figura 13.

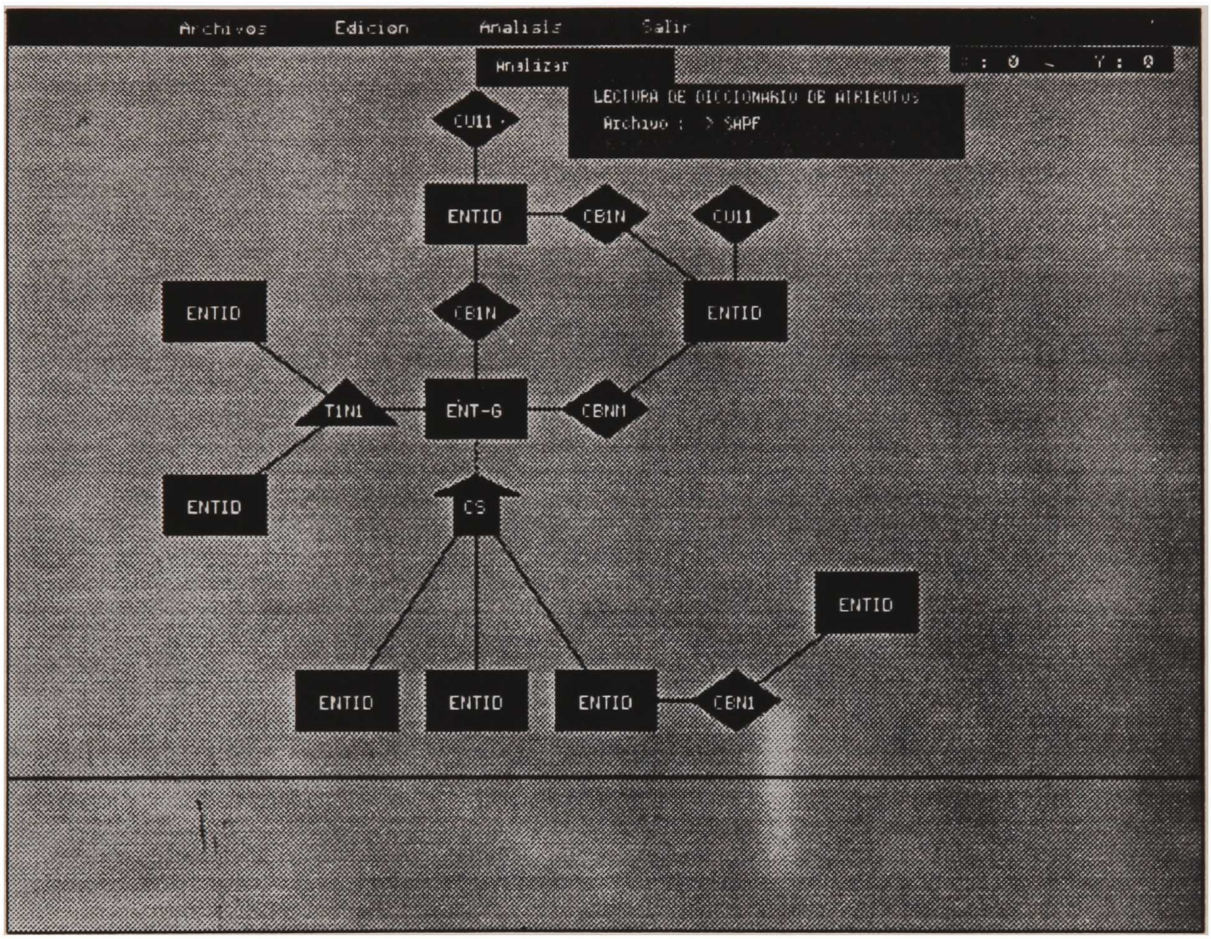


Figura 13: Análisis Sintáctico de un diagrama de Chen.

Una vez que no existe ningún error en la edición del diagrama de Chen se genera un fuente con la extensión *.CHN que sirve de base para el módulo de traducción en nuestro caso el fuente tiene como nombre **SAPF.CHN**, que no está representado en un archivo de texto, por lo tanto no lo presentamos en los anexos.

8.3. TRADUCTOR

El intérprete de traducción se encarga de analizar que el diagrama EVE esté construido correctamente y de la transformación de dicho diagrama al modelo relacional de Codd. Para ésto se emplea el algoritmo de traducción propuesto. El interprete de traducción es accesado a través de icono '**Traductor**' desde la ventana de trabajo EVE en Windows, para usarlo únicamente se introduce el nombre de la base de datos que se desea traducir que en nuestro ejemplo se refiere al SAPF tal como se muestra en la figura 14, una vez introducido el nombre el traductor busca un archivo con terminación *.CHN que sirve de entrada al proceso de traducción. El archivo de entrada (SAPF.CHN) deberá cumplir previamente el análisis sintáctico ya que si contiene algún error el traductor marcará error en el archivo de entrada.

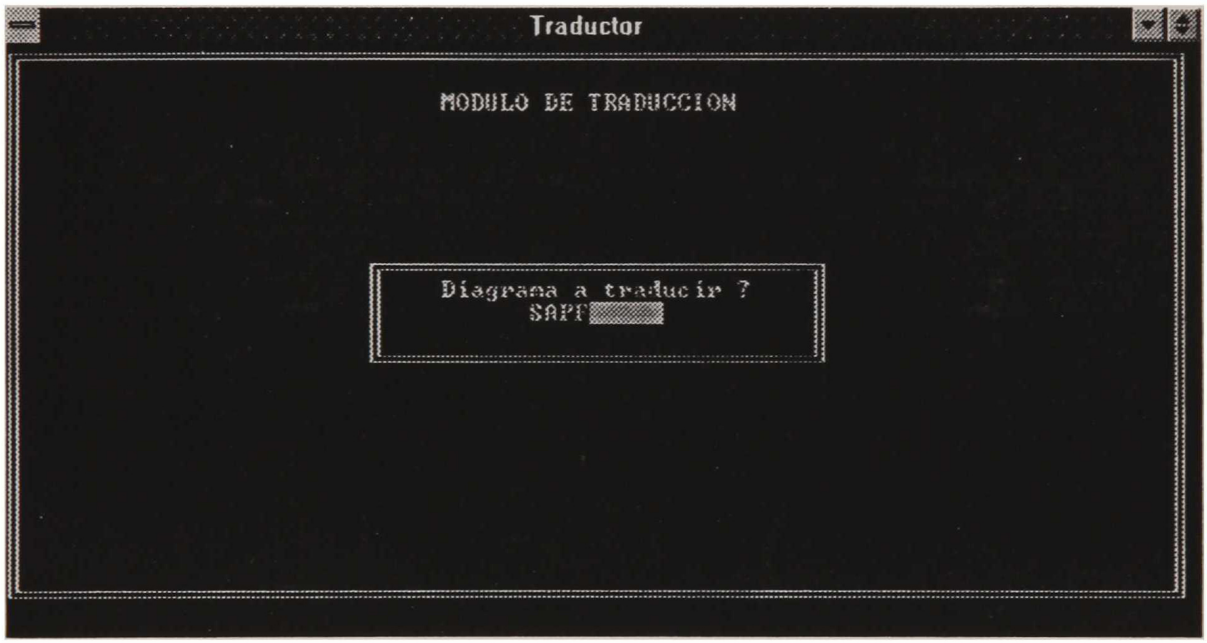


Figura 14: Acceso al traductor de diagramas de Chen a relaciones de Codd.

El traductor produce dos archivos de salida con el mismo nombre de entrada pero con las extensiones *.REL y *.LST. En el ejemplo los archivos de salida son SAPF.LST el cual contiene toda la historia de la traducción; es decir, el detalle de todos los pasos que se realizaron durante la aplicación del algoritmo de traducción y el archivo SAPF.REL el cual contiene el resultado definitivo de la traducción y esta representado en la gramática descrita en el módulo de traducción para representar las relaciones de Codd, este último archivo sirve de entrada para el módulo de Normalización que en el siguiente módulo se describe.

A continuación presentamos la historia de la traducción del SAPF en notación de Codd, donde los atributos llaves aparecen subrayados y los atributos que se van heredando aparecen en letras cursivas.

Paso 1: Definición de Entidades

- **PORTAFOLIOS** (CLAPOR, NUMREG, VALUNI, FECALT, POSICI, TIPLIQ, INTCUS)
- **GP_FINANCIERO** (CLAGRU, RFC, SECTOR, FECALT, ACCEMI, VALLIB)

- **FONDOS** (NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, FIDUCI, REPRES)
- **SUCURSALES** (NUMSUC, NOMBRE, DIRECC, CODPOS, RFC, CUSTOD)
- **CLIENTES** (NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI)
- **NACIONAL** (NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, COMISI, RFC, CLAFIS, MANCUE)
- **INTERNACIONAL**(NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, TIPMON, NACION, CUSVAL, MANCUE)
- **PROMOTORES** (DIGITO, NUMAPO, FECING, NOMBRE, DIRECC, CODPOS, REGLEG, RFC)
- **GESTORES** (NUMGES, REGLEG, NOMBRE, DIRECC, CODPOS, RFC, FECALT)

Paso 2: Relaciones Binarias 1 a 1

- **PORTAFOLIOS** (CLAPOR, NUMREG, VALUNI, FECALT, POSICI, TIPLIQ, INTCUS)
- **GP_FINANCIERO** (CLAGRU, RFC, SECTOR, FECALT, ACCEMI, VALLIB)
- **FONDOS** (NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, FIDUCI, REPRES)
- **SUCURSALES** (NUMSUC, NOMBRE, DIRECC, CODPOS, RFC, CUSTOD)
- **CLIENTES** (NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI)
- **NACIONAL** (NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, COMISI, RFC, CLAFIS, MANCUE)
- **INTERNACIONAL**(NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, TIPMON, NACION, CUSVAL, MANCUE)
- **PROMOTORES** (DIGITO, NUMAPO, FECING, NOMBRE, DIRECC, CODPOS, REGLEG, RFC)
- **GESTORES** (NUMGES, REGLEG, NOMBRE, DIRECC, CODPOS, RFC, FECALT)

Nota : Las relaciones de Codd creadas no sufren ningún efecto debido a que no existen relaciones binarias 1 a 1

Paso 3.1. Relaciones binarias 1 a muchos

- **PORTAFOLIOS** (CLAPOR, NUMREG, VALUNI, FECALT, POSICI, TIPLIQ, INTCUS)
- **GP_FINANCIERO** (CLAGRU, RFC, SECTOR, FECALT, ACCEMI, VALLIB)
- **FONDOS** (NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, FIDUCI, REPRES)
- **SUCURSALES** (NUMSUC, NOMBRE, DIRECC, CODPOS, RFC, CUSTOD)
- **CLIENTES** (NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, ***NUMSUC***)
- **NACIONAL** (NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, COMISI, RFC, CLAFIS, MANCUE)
- **INTERNACIONAL**(NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, TIPMON, NACION, CUSVAL, MANCUE)
- **PROMOTORES** (DIGITO, NUMAPO, FECING, NOMBRE, DIRECC, CODPOS, REGLEG, RFC, ***NUMSUC***)
- **GESTORES** (NUMGES, REGLEG, NOMBRE, DIRECC, CODPOS, RFC, FECALT)

Nota : La relación PROMOTORES y la relación CLIENTES heredan la lleve extraña NUMSUC

Paso 3.2: Relaciones binarias muchos a 1

- **PORTAFOLIOS** (CLAPOR, NUMREG, VALUNI, FECALT, POSICI, TIPLIQ, INTCUS)
- **GP_FINANCIERO** (CLAGRU, RFC, SECTOR, FECALT, ACCEMI, VALLIB)
- **FONDOS** (NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, FIDUCI, REPRES)

- **SUCURSALES** (NUMSUC, NOMBRE, DIRECC, CODPOS, RFC, CUSTOD)
- **CLIENTES** (NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, ***NUMSUC***)
- **NACIONAL** (NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, COMISI, RFC, CLAFIS, MANCUE)
- **INTERNACIONAL**(NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, TIPMON, NACION, CUSVAL, MANCUE, ***NUMGES, REGLEG***)
- **PROMOTORES** (DIGITO, NUMAPO, FECING, NOMBRE, DIRECC, CODPOS, REGLEG, RFC, ***NUMSUC***)
- **GESTORES** (NUMGES, REGLEG, NOMBRE, DIRECC, CODPOS, RFC, FECALT)

Nota : La relación INTERNACIONAL hereda la llave extraña NUMGES, REGLEG

Paso 4. Relaciones unarias 1 a 1 y 1 a muchos

- **PORTAFOLIOS** (CLAPOR, NUMREG, VALUNI, FECALT, POSICI, TIPLIQ, INTCUS)
- **GP_FINANCIERO** (CLAGRU, RFC, SECTOR, FECALT, ACCEMI, VALLIB)
- **FONDOS** (NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, FIDUCI, REPRES)
- **SUCURSALES** (NUMSUC, NOMBRE, DIRECC, CODPOS, RFC, CUSTOD, ***NUMSUC_1***)
- **CLIENTES** (NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, ***NUMSUC***)
- **NACIONAL** (NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, COMISI, RFC, CLAFIS, MANCUE)
- **INTERNACIONAL**(NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, TIPMON, NACION, CUSVAL, MANCUE, ***NUMGES, REGLEG***)
- **PROMOTORES** (DIGITO, NUMAPO, FECING, NOMBRE, DIRECC, CODPOS, REGLEG, RFC, ***NUMSUC, DIGITO_1, NUMAPO_1***)
- **GESTORES** (NUMGES, REGLEG, NOMBRE, DIRECC, CODPOS, RFC, FECALT)

Nota : La relación SUCURSALES se genera el atributo NUMSUC_1 a partir de la llave NUMSUC y en la relación PROMOTORES se generan los atributos DIGITO_1 y NUMAPO_1 a partir de DIGITO y NUMAPO

Paso 5: Conectores de Especialización y/o generalización

- **PORTAFOLIOS** (CLAPOR, NUMREG, VALUNI, FECALT, POSICI, TIPLIQ, INTCUS)
- **GP_FINANCIERO** (CLAGRU, RFC, SECTOR, FECALT, ACCEMI, VALLIB)
- **FONDOS** (NUMCLI, FIDUCI, REPRES)
- **SUCURSALES** (NUMSUC, NOMBRE, DIRECC, CODPOS, RFC, CUSTOD, ***NUMSUC_1***)
- **CLIENTES** (NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, ***NUMSUC***)
- **NACIONAL** (NUMCLI, COMISI, RFC, CLAFIS, MANCUE)
- **INTERNACIONAL**(NUMCLI, TIPMON, NACION, CUSVAL, MANCUE, ***NUMGES, REGLEG***)
- **PROMOTORES** (DIGITO, NUMAPO, FECING, NOMBRE, DIRECC, CODPOS, REGLEG, RFC, ***NUMSUC, DIGITO_1, NUMAPO_1***)
- **GESTORES** (NUMGES, REGLEG, NOMBRE, DIRECC, CODPOS, RFC, FECALT)

Nota : En este paso se realiza la especialización de la relación CLIENTES en las relaciones NACIONAL, FONDOS e INTERNACIONAL a partir del atributo TIPCLI, por lo tanto la relación CLIENTES se queda con los atributos que le son comunes a todas las relaciones y las demás relaciones se quedan únicamente con los atributos que le son particulares.

Paso 6: Relaciones Binarias muchos a muchos y relaciones unarias muchos a muchos

- **PORTAFOLIOS** (CLAPOR, NUMREG, VALUNI, FECALT, POSICI, TIPLIQ, INTCUS)
- **GP_FINANCIERO** (CLAGRU, RFC, SECTOR, FECALT, ACCEMI, VALLIB)
- **FONDOS** (NUMCLI, FIDUCI, REPRES)
- **SUCURSALES** (NUMSUC, NOMBRE, DIRECC, CODPOS, RFC, CUSTOD, ***NUMSUC_1***)
- **CLIENTES** (NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, ***NUMSUC***)
- **NACIONAL** (NUMCLI, COMISI, RFC, CLAFIS, MANCUE)
- **INTERNACIONAL**(NUMCLI, TIPMON, NACION, CUSVAL, MANCUE, ***NUMGES, REGLEG***)
- **PROMOTORES** (DIGITO, NUMAPO, FECING, NOMBRE, DIRECC, CODPOS, REGLEG, RFC, ***NUMSUC, DIGITO_1, NUMAPO_1***)
- **GESTORES** (NUMGES, REGLEG, NOMBRE, DIRECC, CODPOS, RFC, FECALT)
- ***MANEJADO_POR (NUMCLI, DIGITO, NUMPO)***

Nota : Se genera la nueva relación MANEJADO_POR que contiene las llaves de las relaciones CLIENTES y PROMOTORES

Paso 7: Relaciones Ternarias

- **PORTAFOLIOS** (CLAPOR, NUMREG, VALUNI, FECALT, POSICI, TIPLIQ, INTCUS)
- **GP_FINANCIERO** (CLAGRU, RFC, SECTOR, FECALT, ACCEMI, VALLIB)
- **FONDOS** (NUMCLI, FIDUCI, REPRES)
- **SUCURSALES** (NUMSUC, NOMBRE, DIRECC, CODPOS, RFC, CUSTOD, ***NUMSUC_1***)
- **CLIENTES** (NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, ***NUMSUC***)
- **NACIONAL** (NUMCLI, COMISI, RFC, CLAFIS, MANCUE)
- **INTERNACIONAL**(NUMCLI, TIPMON, NACION, CUSVAL, MANCUE, ***NUMGES, REGLEG***)
- **PROMOTORES** (DIGITO, NUMAPO, FECING, NOMBRE, DIRECC, CODPOS, REGLEG, RFC, ***NUMSUC, DIGITO_1, NUMAPO_1***)
- **GESTORES** (NUMGES, REGLEG, NOMBRE, DIRECC, CODPOS, RFC, FECALT)
- ***MANEJADO_POR (NUMCLI, DIGITO, NUMPO)***
- ***OPERA_EN (CLAPOR, NUMREG, CLAGRU, NUMCLI)***

Nota : Se genera la nueva relación OPERA_EN a través de las relaciones PORTAFOLIOS, GP_FINACIERO y CLIENTES

La historia de toda la traducción anterior se genera en el archivo SAPF.LST y se muestra en el **Anexo 3**. Por último cabe señalar que el archivo de salida SAPF.REL que se muestra en el **Anexo 4** contiene las relaciones de Codd resultantes de la traducción tal como se muestran en el Paso 6 de la historia de la traducción del SAPF, pero descritas en lenguaje mostrado en el módulo de traducción además de una relación de los atributos generados durante los pasos con relaciones unarias.

8.4 NORMALIZADOR

El Normalizador tomando como entrada un archivo resultado de la traducción de un diagrama de Chen a relaciones de Codd sin normalizar, se encarga de analizar todas las relaciones funcionales entre los atributos de cada relación y en caso de que el usuario introduzca dependencias funcionales adicionales (que violen la segunda o la tercera forma normal) de normalizar hasta la tercera forma normal la relación de Codd analizada. El Normalizador es accedido a través de icono '**Normalizador**' desde la ventana de trabajo EVE en Windows. Para usarlo, únicamente se introduce el nombre de la base de datos que se desea normalizar (en nuestro ejemplo se refiere al SAPF) tal como se muestra en la figura 15. Una vez introducido el nombre el normalizador busca un archivo con terminación *.REL que sirve de entrada al proceso de normalización. El archivo de entrada (SAPF.REL) deberá ser producto de la traducción de un diagrama de Chen a relaciones de Codd y estar representado en el lenguaje de descripción de relaciones de Codd previamente descrito.

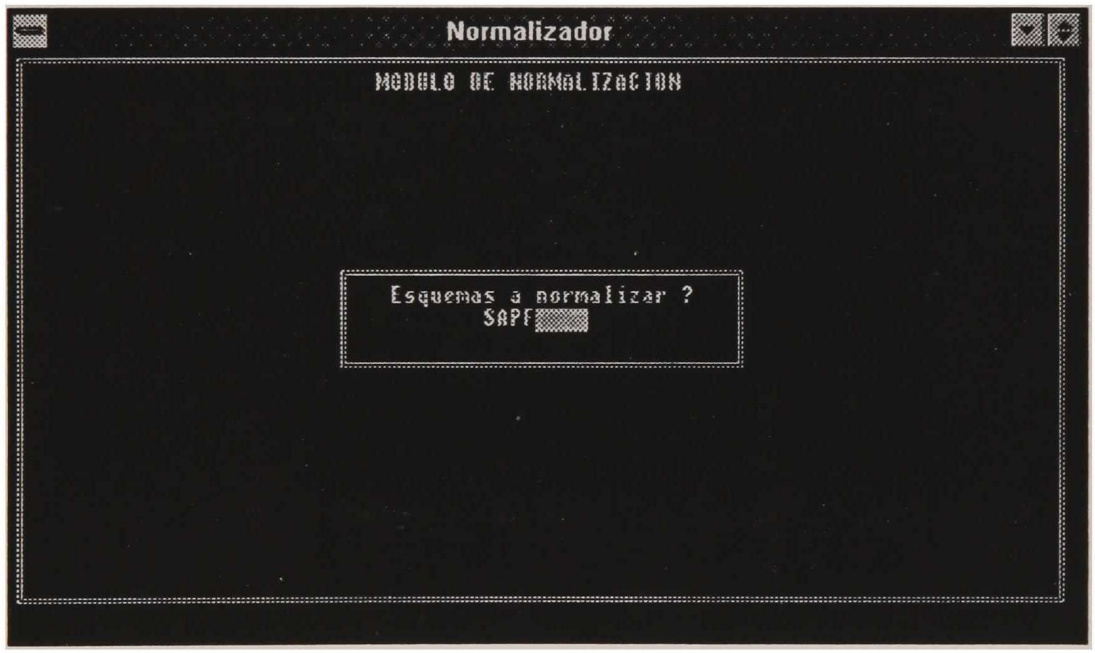


Figura 15: Normalización del Sistema de Administración de Portafolios Financieros (SAPF)

Para el proceso de normalización es necesario que en cualesquiera de las relaciones de Codd traducidas existan dependencias funcionales que violen la segunda o la tercera forma normal. En la base de datos ejemplo SAPF se presentan las siguientes :

- Relación PROMOTORES.- se viola la segunda formal normal ya que el atributo NUMAPO (Número de Apoderado) implica funcionalmente a los atributos FECING (Fecha de Ingreso) y REGLEG (Registro Legal), es decir, NUMAPO -> (FECING, REGLEG) ya que el registro del número de apoderado lleva implícito un registro legal y una fecha de ingreso ante la CNV (Comisión Nacional de Valores). Como NUMAPO es un atributo que forma parte de la llave esta es una clara violación a la segunda forma normal.

- Relación PORTAFOLIOS.- se viola la tercera forma normal ya que el atributo INTCUS (Institución de Custodia) implica funcionalmente al atributo TIPLIQ (Tipo de Liquidación), es decir, INTCUS -> TIPLIQ ya que dependiendo de la Institución que realiza la custodia de los valores es la forma en cómo se debe realizar la

liquidación de los valores. Como INTCUS y TIPLIQ son atributos no-llaves esta es una clara violación a la tercera forma normal.

La forma de operar del Normalizador es la siguiente :

- Una vez introducido el nombre de la base de datos a Normalizar (SAPF.REL) el Normalizador carga una por una las relaciones de Codd a normalizar y presenta una pantalla que contiene el nombre de la relación que se está normalizando y todas las dependencias funcionales que existen para esa relación. Para consultar las dependencias funcionales existentes se pueden utilizar las letras **S** la cual muestra la dependencia funcional siguiente o **A** la cual muestra la dependencia funcional anterior. En la figura 16 se muestra este paso del Normalizador con la relación PORTAFOLIOS, en este caso se muestra la dependencia funcional CLAPOR, NUMREG -> VALUNI.

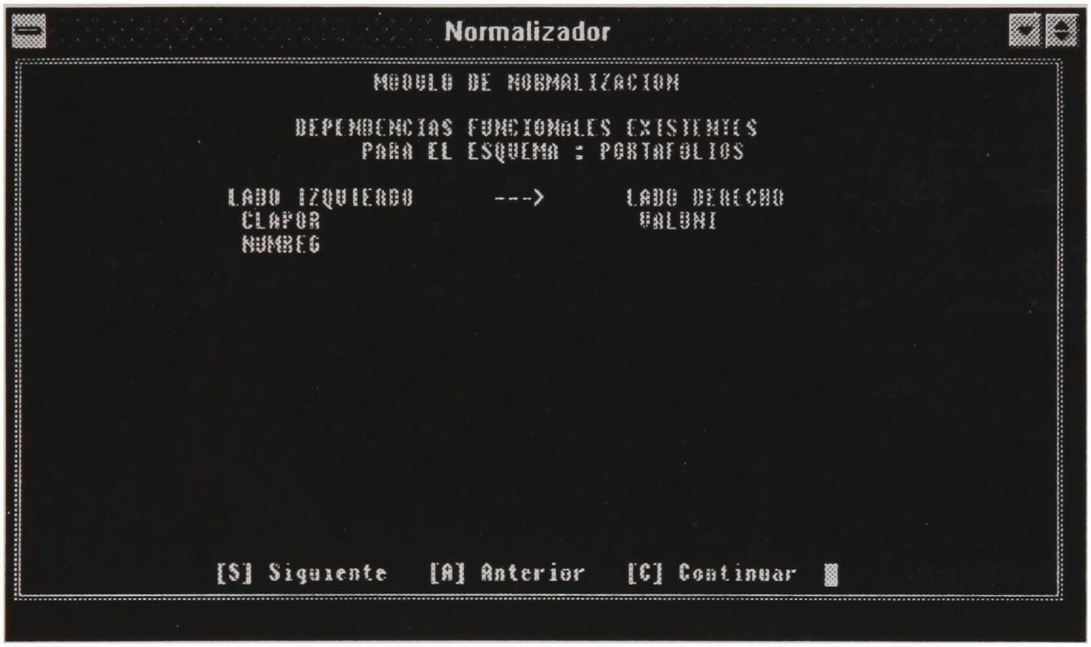


Figura 16: Presentación de las dependencias funcionales del SAPF.

- Para introducir alguna dependencia funcional adicional en la pantalla anterior será necesario Usar la letra **C** de continuar y contestar el mensaje de si se desea agregar dependencias funcionales adicionales en la figura 17 se muestra la manera de como en la relación PORTAFOLIOS se introduce la dependencia funcional INTCUS -> TIPLIQ.

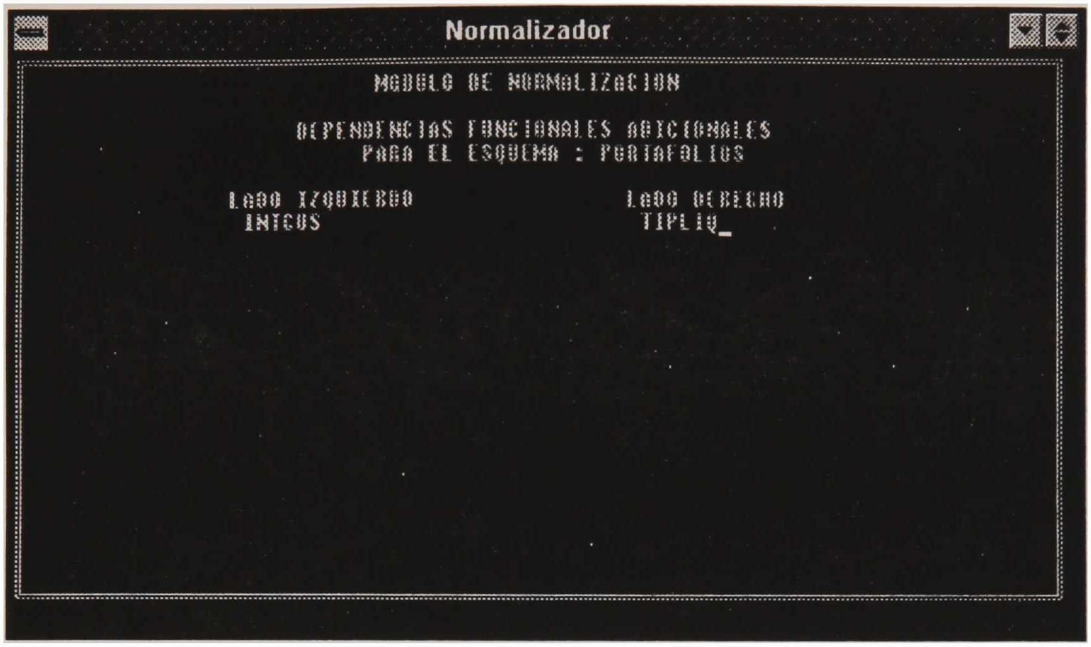


Figura 17: Incorporación de dependencias funcionales adicionales

- Por último, una vez introducidas las dependencias funcionales adicionales, el Normalizador pregunta si se desean consultar las dependencias funcionales existentes (incluyendo las capturadas) para la relación que se esta normalizando, en caso de no querer consultar se realiza la normalización a tercera forma normal de la relación en curso y se procede a cargar la siguiente relación para ejecutar el mismo análisis.

Una vez que el Normalizador a completado su tarea genera un archivo de salida con terminación *.3FN (SAPF.3FN) que contiene la base de datos con todas sus relaciones en tercera forma normal. Las relaciones en tercera forma normal están representadas en el lenguaje descrito en el módulo de normalización y se detallan en el **Anexo 5**.

Unicamente de forma descriptiva señalaremos como el producto de la Normalización que la Relación PORTAFOLIOS se partió en dos relaciones en tercera forma normal PORTAFOLIOS_a y PORTAFOLIOS_b y la relación PROMOTORES se partió en dos relaciones en tercera forma normal PROMOTORES_a y PROMOTORES_b de tal suerte que la base de datos del SAPF en tercera forma normal quedó representada tal como se muestra a continuación :

- **PORTAFOLIOS_a (CLAPOR, NUMREG, VALUNI, FECALT, POSICI, INTCUS)**
- ***PORTAFOLIOS_b (INTCUS, TIPLIQ)***
- GP_FINANCIERO (CLAGRU, RFC, SECTOR, FECALT, ACCEMI, VALLIB)
- FONDOS (NUMCLI, FIDUCI, REPRES)
- SUCURSALES (NUMSUC, NOMBRE, DIRECC, CODPOS, RFC, CUSTOD, NUMSUC_1)
- CLIENTES (NUMCLI, TIPCLI, PERJUR, NOMBRE, DIRECC, CODPOS, VALCAR, EFECTI, NUMSUC)
- NACIONAL (NUMCLI, COMISI, RFC, CLAFIS, MANCUE)
- INTERNACIONAL(NUMCLI, TIPMON, NACION, CUSVAL, MANCUE, NUMGES, REGLEG)

- **PROMOTORES_a (DIGITO,NUMAPO, NOMBRE, DIRECC, CODPOS, RFC, NUMSUC,** DIGITO_1, NUMAPO_1)
- ***PROMOTORES_b (NUMAPO, FECING, REGLEG)***
- GESTORES (NUMGES, REGLEG, NOMBRE, DIRECC, CODPOS, RFC, FECALT)
- MANEJADO_POR (NUMCLI, DIGITO, NUMPO)
- OPERA_EN (CLAPOR, NUMREG, CLAGRU, NUMCLI)

8.5 DISEÑO FISICO

El diseño físico de la base de datos es el proceso por cual se escriben las especificaciones de la base de datos en el lenguaje definido por cada Manejador de Base de Datos (En nuestro caso el manejador definido para LIDA). El Diseño Físico es accedido a través de icono '**Diseño Físico**' desde la ventana de trabajo EVE en Windows. Su uso es bastante sencillo y consiste básicamente en especificar al sistema el nombre de la base de datos recién normalizada y cuya extensión por omisión es (*.3FN) y el nombre del Diccionario de atributos, ya compilado, que sirve de base para la descripción física de la base de datos.

Una vez que se digita el icono de Diseño Físico el sistema presenta una ventana donde se solicita se incorpore el nombre del diccionario de atributos que servirá de base para la creación de la base de datos física tal como se muestra en la figura 18.

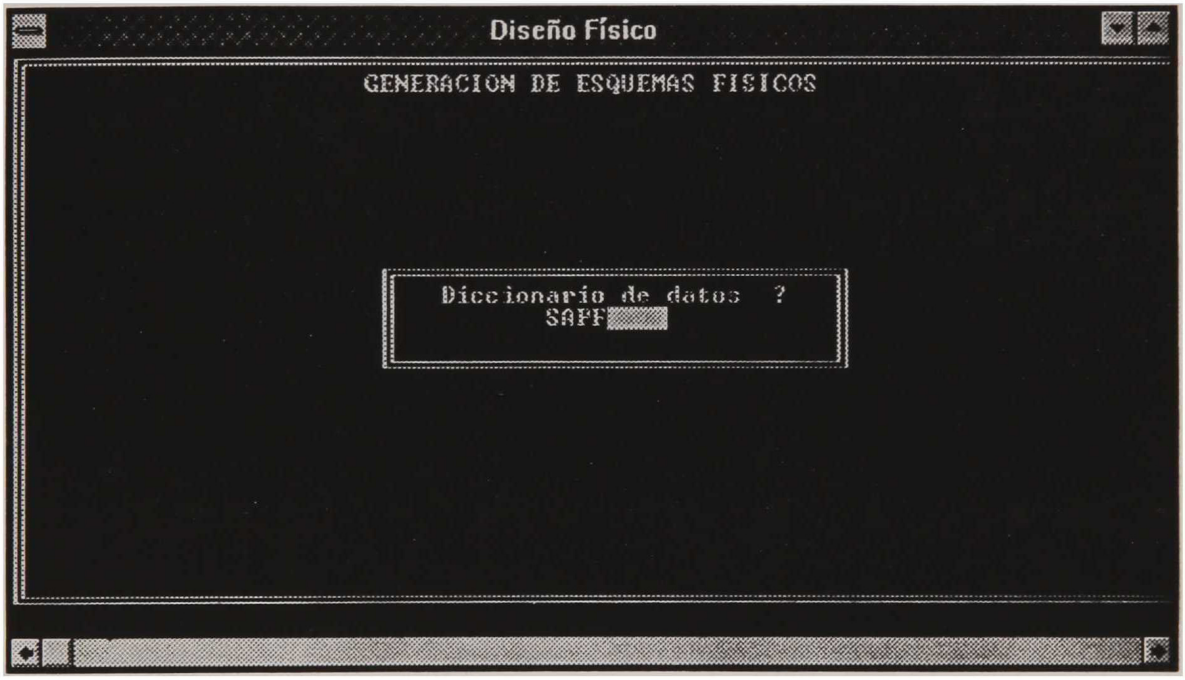


Figura 18: Introducción del diccionario de atributos para el proceso de diseño físico

A continuación el sistema presenta una ventana donde se solicita se incorpore el nombre de la base de datos en tercera forma normal (archivo con extensión *.3FN) que servirá de base para la creación de la base de datos física tal como se muestra en la figura 19.

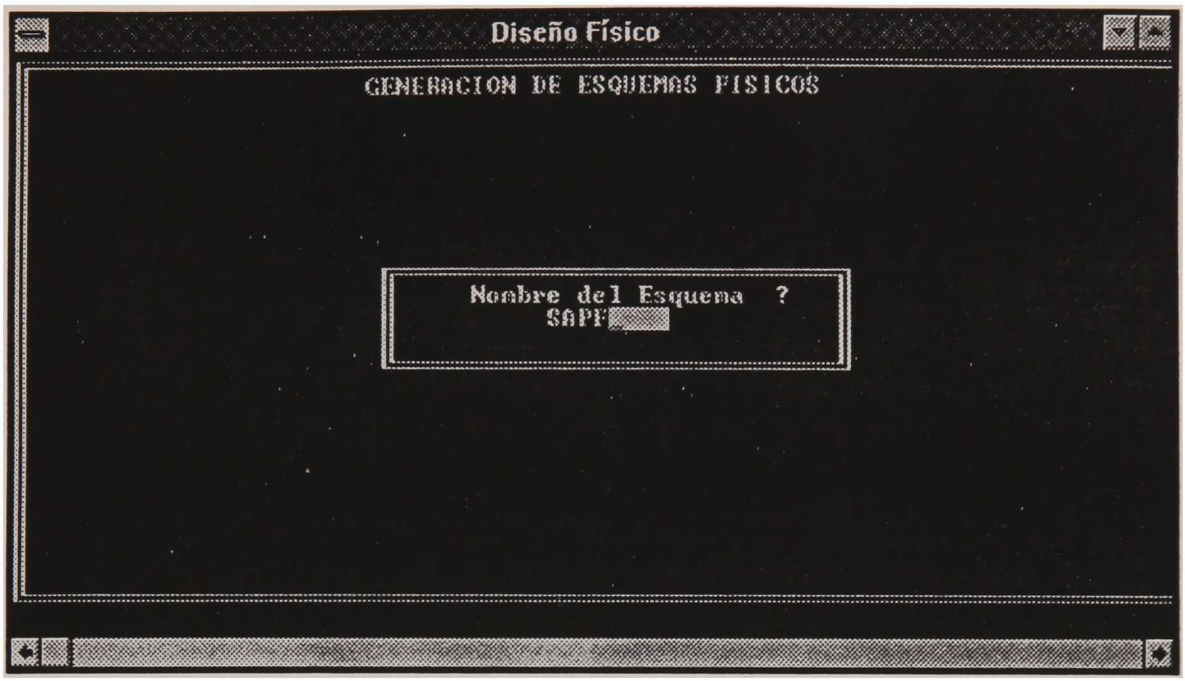


Figura 19: Introducción del archivo en tercera forma normal para el proceso de diseño físico

Como resultado del proceso de diseño físico se genera el diseño físico de la base de datos en la especificación requerida por el manejador de bases de datos desarrollado para LIDA en el CINVESTAV tal como se muestra en el Anexo 6. Cabe señalar que el diseño físico no está limitado para el manejador LIDA, ya que contando con el diccionario de atributos y con las relaciones de Codd en tercera forma normal es factible realizar el diseño físico para cualquier manejador de base de datos que se desee, para lo cual será necesario investigar con detalle la sintaxis de cada manejador de base de datos donde se debe realizar el diseño físico y describirlo en base a la misma.

9. CONCLUSIONES

Del presente trabajo se pueden derivar dos conclusiones fundamentales, la primera se refiere a la arquitectura del SDBD-EVE y la segunda se refiere a la funcionalidad del SDBD-EVE respecto al diseño de base de datos físicas.

La arquitectura del SDBD-EVE se basa en los fundamentos que sobre programación visual expresó el doctor Sergio Chapa en la definición de LIDA [CHAP91], pero a diferencia de la definición de LIDA los diagramas de Chen utilizados para el diseño de base de datos no son diagramas orientados. El diccionario de atributos del SDBD es un diccionario central, cuya definición se dirige por sintaxis, lo cual permite tener una definición consistente del repositorio de datos, es factible realizar modificaciones al diccionario de atributos como la incorporación de validaciones, características de despliegue y documentación realizando pequeños cambios en la gramática del diccionario. La edición de diagramas de Chen es dirigida por sintaxis basada en la gramática de construcción propuesta. La traducción de diagramas de Chen a relaciones de Codd que inicialmente se pensó realizar dirigida por sintaxis se realizó vía el algoritmo de interpretación propuesto. No se realizó ninguna prueba formal sobre la completitud del algoritmo de interpretación pero se realizaron pruebas parciales con ejemplos los suficientemente versátiles que incorporaron varios esquemas de base de datos con buenos resultados. La normalización de la base de datos se basó en el algoritmo de Bernstein tomando como base un trabajo sobre normalización de base de datos

en C previamente desarrollado [HERN90] con una alta funcionalidad. El proceso de diseño físico de la base de datos esta orientado al manejador de base de datos utilizado por LIDA.

La funcionalidad de SDBD-EVE es alta ya que al estar montado sobre Windows permite trabajar con las cinco herramientas (Diccionario de atributos, Editor de Chen, Normalizador, Traductor, Diseño Físico) de forma independiente, inclusive permite la edición de varios diagramas de Chen al mismo tiempo. En la descripción de SDBD-EVE se utilizó como ejemplo de aplicación el Sistema para Administración de Portafolios Financieros (SAPF), el cual es lo suficientemente completo para mostrar todas las facilidades que proporciona SDBD. La base de datos del SAPF tiene la suficiente complejidad para mostrar todos los posibles conectores que proporciona la implementación realizada del modelo de Chen. Por último cabe señalar que SDBD-EVE cumple con todos los requisitos de una herramienta CASE para el diseño de base de datos ya que incorpora elementos que van desde la creación del repositorio de datos hasta el diseño físico de la base de datos

10. LITERATURA CITADA

- [BEER78] Beeri, C. and Bernstein, P. A: and Goodman, N. A sophisticate's introduction to databasenormalization theory, In Procceding of the 4th International Conference on Very Large Data bases (West Berlin)., 1978. pp. 113-124.

- [BERN76] Bernstein, P. A: Synthesizing third normal form relations from functional dependencies. ACM Trans. on Database Syst. 1,4., 1976. pp. 277-298

- [CHAP85] Chapa V. S. Herramienta para consulta y captura basadas en el descriptor de Archivo. Reporte técnico No. 15, Serie amarilla investigación, Departamento de Ingeniería Electrica, CINVESTAV-IPN. 1985.

- [CHAP91] Chapa V. S. Programación automática a partir de descritores de flujo de información. Tesis Doctoral. Departamento de Ingeniería Electrica, CINVESTAV-IPN México. 1991.

- [CHEN76] Chen, P. The entity-relationship model. Toward a unified view of data. ACM Trans. Datbase Syst. 1,1., 1976. pp. 9-36

- [DIED88] Diederich, J. and Milton, J. New Methods and Fast Algoritms for Data Base Normalization. ACM Trans. on Datbase Syst. 13, 3., 1988. pp.339-365.

- [FAGI77] Fagin, R. Multivalued Dependencies and A New Normal Form for Relational Databases. ACM Trans. on Datbase Syst. 2, 3., 1977. pp.262-278.

- [IBM89] IBM. AD/cycle., 1989. 43 p.

- [HERN90] Hernández S.R y Chapa V. S. Normalización de base de datos en C. Reporte técnico No.107, Serie amarilla investigación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN. 1990.

- [MAIE83] Maier, D. The Theory of Relational Databases. computer Science Press, Rockville, Md. 1983

- [RISS77] Rissanen, J. Independent components os Relations. AC. Trans. on Database Syst. 2, 4., 1977. pp. 317-325

- [TEOR86] Teorey, J. T. A logical desing methodology for relational databases using the extended entity-relationship model. Computing Surveys, Vol. 18 No. 2., 1986. pp. 197-222.

11. ANEXOS

ANEXO 1: DEFINICION DEL DICCIONARIO DE ATRIBUTOS

```
;-----------------------------------------------------------------
;DICCIONARIO DE ATRIBUTOS DE LA BASE DE DATOS DE PORTAFOLIOS
;-----------------------------------------------------------------

#diccionario
CLAPOR, A, 7, "_______"
VALUNI, N, 10, "__________"
FECALT, D, 6, "__/__/__"
POSICI, N, 15, "_______________"
TIPLIQ, A, 2, "__"
INTCUS, A, 40, "________________________________________"
NUMREG, Z, 7, "_______"
DIGITO, A, 2, "__"
NUMAPO, N, 10, "__________"
FECING, D, 6, "__/__/__"
NOMBRE, A, 60, "____________________________________________________________"
DIRECC, A, 60, "____________________________________________________________"
CODPOS, Z, 5, "_____"
REGLEG, A, 40, "________________________________________"
RFC,   A, 14, "______________"
CLAGRU, A, 7, "_______"
SECTOR, A, 2, "__"
ACCEMI, N, 15, "_______________"
VALLIB, N, 10, "__________"
NUMCLI, Z, 7, "_______"
TIPCLI, A, 2, "__"
PERJUR, A, 2, "__"
VALCAR, C, 15, "$_____.__"
EFECTI, C, 15, "$_____.__"
TIPMON, A, 2, "__"
NACION, A, 40, "________________________________________"
CUSVAL, A, 40, "________________________________________"
MANCUE, A, 2, "__"
COMISI, N, 2, "%__"
CLAFIS, A, 5, "_____"
FIDUCI, A, 40, "________________________________________"
REPRES, A, 60, "____________________________________________________________"
NUMGES, Z, 5, "_____"
NUMSUC, Z, 5, "_____"
CUSTOD, C, 15, "$_____.__"
;-----------------------------------------------------------------
;LA DESCRIPCION DE LOS CAMPOS DEL DICCIONARIO SE PRESENTA A CONTINUACION
;-----------------------------------------------------------------
;Clave del portafolio
;Valor unitario
;Fecha de alta
;Posicion
;Tipo de liquidacion
```

;Numero de Registro
;Digito o clave del promotor
;Numero de apoderado legal
;Fecha de ingreso
;Nombre
;Direccion
;Codigo postal
;Registro legal
;Registro Federal de Causantes
;Clave de Grupo
;Sector Economico
;Acciones Emitidas
;Valor en libros
;Numero de cliente
;Tipo de cliente
;Personalidad Jurdica
;Valor de Cartera
;Efectivo a la Fecha
;Tipo de moneda
;Nacionalidad
;Custodio de valores
;Manejo de la cuenta
;Porcentaje de comision
;Clave fiscal
;Fiduciario
;Representante
;Numero de gestor
;Numero de sucursal
;Custodia de valores
#fin diccionario
;----------
;----------

ANEXO 2: DICCIONARIO DE ATRIBUTOS (*.DIC)

```
CLAPOR,A,7,"_______"
VALUNI,N,10,"__________"
FECALT,D,6,"__/__/__"
POSICI,N,15,"_______________"
TIPLIQ,A,2,"__"
INTCUS,A,40,"________________________________________"
NUMREG,Z,7,"_______"
DIGITO,A,2,"__"
NUMAPO,N,10,"__________"
FECING,D,6,"__/__/__"
NOMBRE,A,60,"____________________________________________________________"
DIRECC,A,60,"____________________________________________________________"
CODPOS,Z,5,"_____"
REGLEG,A,40,"________________________________________"
RFC,A,14,"______________"
CLAGRU,A,7,"_______"
SECTOR,A,2,"__"
ACCEMI,N,15,"_______________"
VALLIB,N,10,"__________"
NUMCLI,Z,7,"_______"
TIPCLI,A,2,"__"
PERJUR,A,2,"__"
VALCAR,C,15,"$_____.__"
EFECTI,C,15,"$_____.__"
TIPMON,A,2,"__"
NACION,A,40,"________________________________________"
CUSVAL,A,40,"________________________________________"
MANCUE,A,2,"__"
COMISI,N,2,"%__"
CLAFIS,A,5,"_____"
FIDUCI,A,40,"________________________________________"
REPRES,A,60,"____________________________________________________________"
NUMGES,Z,5,"_____"
NUMSUC,Z,5,"_____"
CUSTOD,C,15,"$_____.__"
```

ANEXO 3: TRADUCCION DEL SISTEMA DE ADMINISTRACION DE PORTAFOLIOS FINANCIEROS (SAPF) A RELACIONES DE CODD

Paso Entidades:

Nombre: PORTAFOLIOS

Llave: CLAPOR
Llave: NUMREG

Descriptor: VALUNI
Descriptor: FECALT
Descriptor: POSICI
Descriptor: TIPLIQ
Descriptor: INTCUS

Nombre: GP_FINANCIERO

Llave: CLAGRU

Descriptor: RFC
Descriptor: SECTOR
Descriptor: FECALT
Descriptor: ACCEMI
Descriptor: VALLIB

Nombre: FONDOS

Llave: NUMCLI

Descriptor: TIPCLI
Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: VALCAR
Descriptor: EFECTI
Descriptor: FIDUCI
Descriptor: REPRES

Nombre: SUCURSALES

Llave: NUMSUC

Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: RFC
Descriptor: CUSTOD

Nombre: CLIENTES

Llave: NUMCLI

Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: VALCAR
Descriptor: EFECTI
Descriptor: TIPCLI

Nombre: NACIONAL

Llave: NUMCLI

Descriptor: TIPCLI
Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: VALCAR
Descriptor: EFECTI
Descriptor: COMISI
Descriptor: RFC
Descriptor: CLAFIS
Descriptor: MANCUE

Nombre: INTERNACIONAL

Llave: NUMCLI

Descriptor: TIPCLI
Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: EFECTI
Descriptor: TIPMON
Descriptor: NACION
Descriptor: CUSVAL
Descriptor: MANCUE
Descriptor: VALCAR

Nombre: PROMOTORES

Llave: DIGITO
Llave: NUMAPO

Descriptor: FECING
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: REGLEG
Descriptor: RFC

Nombre: GESTORES

Llave: NUMGES
Llave: REGLEG

Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: RFC
Descriptor: FECALT

Paso Bin 11:

Nombre: PORTAFOLIOS

Llave: CLAPOR
Llave: NUMREG

Descriptor: VALUNI
Descriptor: FECALT
Descriptor: POSICI
Descriptor: TIPLIQ
Descriptor: INTCUS

Nombre: GP_FINANCIERO

Llave: CLAGRU

Descriptor: RFC
Descriptor: SECTOR
Descriptor: FECALT
Descriptor: ACCEMI
Descriptor: VALLIB

Nombre: FONDOS

Llave: NUMCLI

Descriptor: TIPCLI
Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: VALCAR
Descriptor: EFECTI
Descriptor: FIDUCI
Descriptor: REPRES

Nombre: SUCURSALES

Llave: NUMSUC

Descriptor: NOMBRE

Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: RFC
Descriptor: CUSTOD

Nombre: CLIENTES

Llave: NUMCLI

Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: VALCAR
Descriptor: EFECTI
Descriptor: TIPCLI

Nombre: NACIONAL

Llave: NUMCLI

Descriptor: TIPCLI
Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: VALCAR
Descriptor: EFECTI
Descriptor: COMISI
Descriptor: RFC
Descriptor: CLAFIS
Descriptor: MANCUE

Nombre: INTERNACIONAL

Llave: NUMCLI

Descriptor: TIPCLI
Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: EFECTI
Descriptor: TIPMON
Descriptor: NACION
Descriptor: CUSVAL
Descriptor: MANCUE
Descriptor: VALCAR

Nombre: PROMOTORES

Llave: DIGITO
Llave: NUMAPO

Descriptor: FECING
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: REGLEG
Descriptor: RFC

Nombre: GESTORES

Llave: NUMGES
Llave: REGLEG

Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: RFC
Descriptor: FECALT

Paso Bin 1m:

Nombre: PORTAFOLIOS

Llave: CLAPOR
Llave: NUMREG

Descriptor: VALUNI
Descriptor: FECALT
Descriptor: POSICI
Descriptor: TIPLIQ
Descriptor: INTCUS

Nombre: GP_FINANCIERO

Llave: CLAGRU

Descriptor: RFC
Descriptor: SECTOR
Descriptor: FECALT
Descriptor: ACCEMI
Descriptor: VALLIB

Nombre: FONDOS

Llave: NUMCLI

Descriptor: TIPCLI
Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: VALCAR
Descriptor: EFECTI

Descriptor: FIDUCI
Descriptor: REPRES

Nombre: SUCURSALES

Llave: NUMSUC

Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: RFC
Descriptor: CUSTOD

Nombre: CLIENTES

Llave: NUMCLI

Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: VALCAR
Descriptor: EFECTI
Descriptor: TIPCLI
Descriptor: NUMSUC

Nombre: NACIONAL

Llave: NUMCLI

Descriptor: TIPCLI
Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: VALCAR
Descriptor: EFECTI
Descriptor: COMISI
Descriptor: RFC
Descriptor: CLAFIS
Descriptor: MANCUE

Nombre: INTERNACIONAL

Llave: NUMCLI

Descriptor: TIPCLI
Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: EFECTI
Descriptor: TIPMON
Descriptor: NACION

Descriptor: CUSVAL
Descriptor: MANCUE
Descriptor: VALCAR

Nombre: PROMOTORES

Llave: DIGITO
Llave: NUMAPO

Descriptor: FECING
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: REGLEG
Descriptor: RFC
Descriptor: NUMSUC

Nombre: GESTORES

Llave: NUMGES
Llave: REGLEG

Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: RFC
Descriptor: FECALT

Paso Bin m1:

Nombre: PORTAFOLIOS

Llave: CLAPOR
Llave: NUMREG

Descriptor: VALUNI
Descriptor: FECALT
Descriptor: POSICI
Descriptor: TIPLIQ
Descriptor: INTCUS

Nombre: GP_FINANCIERO

Llave: CLAGRU

Descriptor: RFC
Descriptor: SECTOR
Descriptor: FECALT
Descriptor: ACCEMI
Descriptor: VALLIB

Nombre: FONDOS

Llave: NUMCLI

Descriptor: TIPCLI
Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: VALCAR
Descriptor: EFECTI
Descriptor: FIDUCI
Descriptor: REPRES

Nombre: SUCURSALES

Llave: NUMSUC

Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: RFC
Descriptor: CUSTOD

Nombre: CLIENTES

Llave: NUMCLI

Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: VALCAR
Descriptor: EFECTI
Descriptor: TIPCLI
Descriptor: NUMSUC

Nombre: NACIONAL

Llave: NUMCLI

Descriptor: TIPCLI
Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: VALCAR
Descriptor: EFECTI
Descriptor: COMISI
Descriptor: RFC
Descriptor: CLAFIS
Descriptor: MANCUE

Nombre: INTERNACIONAL

Llave: NUMCLI

Descriptor: TIPCLI
Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: EFECTI
Descriptor: TIPMON
Descriptor: NACION
Descriptor: CUSVAL
Descriptor: MANCUE
Descriptor: VALCAR
Descriptor: NUMGES
Descriptor: REGLEG

Nombre: PROMOTORES

Llave: DIGITO
Llave: NUMAPO

Descriptor: FECING
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: REGLEG
Descriptor: RFC
Descriptor: NUMSUC

Nombre: GESTORES

Llave: NUMGES
Llave: REGLEG

Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: RFC
Descriptor: FECALT

Paso Una 11 y 1m:

Nombre: PORTAFOLIOS

Llave: CLAPOR
Llave: NUMREG

Descriptor: VALUNI
Descriptor: FECALT
Descriptor: POSICI
Descriptor: TIPLIQ
Descriptor: INTCUS

Nombre: GP_FINANCIERO

Llave: CLAGRU

Descriptor: RFC
Descriptor: SECTOR
Descriptor: FECALT
Descriptor: ACCEMI
Descriptor: VALLIB

Nombre: FONDOS

Llave: NUMCLI

Descriptor: TIPCLI
Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: VALCAR
Descriptor: EFECTI
Descriptor: FIDUCI
Descriptor: REPRES

Nombre: SUCURSALES

Llave: NUMSUC

Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: RFC
Descriptor: CUSTOD
Descriptor: NUMSUC_1

Nombre: CLIENTES

Llave: NUMCLI

Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: VALCAR
Descriptor: EFECTI
Descriptor: TIPCLI
Descriptor: NUMSUC

Nombre: NACIONAL

Llave: NUMCLI

Descriptor: TIPCLI

Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: VALCAR
Descriptor: EFECTI
Descriptor: COMISI
Descriptor: RFC
Descriptor: CLAFIS
Descriptor: MANCUE

Nombre: INTERNACIONAL

Llave: NUMCLI

Descriptor: TIPCLI
Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: EFECTI
Descriptor: TIPMON
Descriptor: NACION
Descriptor: CUSVAL
Descriptor: MANCUE
Descriptor: VALCAR
Descriptor: NUMGES
Descriptor: REGLEG

Nombre: PROMOTORES

Llave: DIGITO
Llave: NUMAPO

Descriptor: FECING
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: REGLEG
Descriptor: RFC
Descriptor: NUMSUC
Descriptor: DIGITO_1
Descriptor: NUMAPO_1

Nombre: GESTORES

Llave: NUMGES
Llave: REGLEG

Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: RFC
Descriptor: FECALT

Paso Con gen esp:

Nombre: PORTAFOLIOS

Llave: CLAPOR
Llave: NUMREG

Descriptor: VALUNI
Descriptor: FECALT
Descriptor: POSICI
Descriptor: TIPLIQ
Descriptor: INTCUS

Nombre: GP_FINANCIERO

Llave: CLAGRU

Descriptor: RFC
Descriptor: SECTOR
Descriptor: FECALT
Descriptor: ACCEMI
Descriptor: VALLIB

Nombre: FONDOS

Llave: NUMCLI

Descriptor: FIDUCI
Descriptor: REPRES

Nombre: SUCURSALES

Llave: NUMSUC

Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: RFC
Descriptor: CUSTOD
Descriptor: NUMSUC_1

Nombre: CLIENTES

Llave: NUMCLI

Descriptor: TIPCLI
Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: VALCAR

Descriptor: EFECTI
Descriptor: NUMSUC

Nombre: NACIONAL

Llave: NUMCLI

Descriptor: COMISI
Descriptor: RFC
Descriptor: CLAFIS
Descriptor: MANCUE

Nombre: INTERNACIONAL

Llave: NUMCLI

Descriptor: TIPMON
Descriptor: NACION
Descriptor: CUSVAL
Descriptor: MANCUE
Descriptor: NUMGES
Descriptor: REGLEG

Nombre: PROMOTORES

Llave: DIGITO
Llave: NUMAPO

Descriptor: FECING
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: REGLEG
Descriptor: RFC
Descriptor: NUMSUC
Descriptor: DIGITO_1
Descriptor: NUMAPO_1

Nombre: GESTORES

Llave: NUMGES
Llave: REGLEG

Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: RFC
Descriptor: FECALT

Paso Bin mm:

Nombre: PORTAFOLIOS

Llave: CLAPOR
Llave: NUMREG

Descriptor: VALUNI
Descriptor: FECALT
Descriptor: POSICI
Descriptor: TIPLIQ
Descriptor: INTCUS

Nombre: GP_FINANCIERO

Llave: CLAGRU

Descriptor: RFC
Descriptor: SECTOR
Descriptor: FECALT
Descriptor: ACCEMI
Descriptor: VALLIB

Nombre: FONDOS

Llave: NUMCLI

Descriptor: FIDUCI
Descriptor: REPRES

Nombre: SUCURSALES

Llave: NUMSUC

Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: RFC
Descriptor: CUSTOD
Descriptor: NUMSUC_1

Nombre: CLIENTES

Llave: NUMCLI

Descriptor: TIPCLI
Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: VALCAR
Descriptor: EFECTI
Descriptor: NUMSUC

Nombre: NACIONAL

Llave: NUMCLI

Descriptor: COMISI
Descriptor: RFC
Descriptor: CLAFIS
Descriptor: MANCUE

Nombre: INTERNACIONAL

Llave: NUMCLI

Descriptor: TIPMON
Descriptor: NACION
Descriptor: CUSVAL
Descriptor: MANCUE
Descriptor: NUMGES
Descriptor: REGLEG

Nombre: PROMOTORES

Llave: DIGITO
Llave: NUMAPO

Descriptor: FECING
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: REGLEG
Descriptor: RFC
Descriptor: NUMSUC
Descriptor: DIGITO_1
Descriptor: NUMAPO_1

Nombre: GESTORES

Llave: NUMGES
Llave: REGLEG

Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: RFC
Descriptor: FECALT

Nombre: MANEJADO_POR

Llave: NUMCLI
Llave: DIGITO
Llave: NUMAPO

Paso Una mm:

Nombre: PORTAFOLIOS

Llave: CLAPOR
Llave: NUMREG

Descriptor: VALUNI
Descriptor: FECALT
Descriptor: POSICI
Descriptor: TIPLIQ
Descriptor: INTCUS

Nombre: GP_FINANCIERO

Llave: CLAGRU

Descriptor: RFC
Descriptor: SECTOR
Descriptor: FECALT
Descriptor: ACCEMI
Descriptor: VALLIB

Nombre: FONDOS

Llave: NUMCLI

Descriptor: FIDUCI
Descriptor: REPRES

Nombre: SUCURSALES

Llave: NUMSUC

Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: RFC
Descriptor: CUSTOD
Descriptor: NUMSUC_1

Nombre: CLIENTES

Llave: NUMCLI

Descriptor: TIPCLI
Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: VALCAR
Descriptor: EFECTI
Descriptor: NUMSUC

Nombre: NACIONAL

Llave: NUMCLI

Descriptor: COMISI
Descriptor: RFC
Descriptor: CLAFIS
Descriptor: MANCUE

Nombre: INTERNACIONAL

Llave: NUMCLI

Descriptor: TIPMON
Descriptor: NACION
Descriptor: CUSVAL
Descriptor: MANCUE
Descriptor: NUMGES
Descriptor: REGLEG

Nombre: PROMOTORES

Llave: DIGITO
Llave: NUMAPO

Descriptor: FECING
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: REGLEG
Descriptor: RFC
Descriptor: NUMSUC
Descriptor: DIGITO_1
Descriptor: NUMAPO_1

Nombre: GESTORES

Llave: NUMGES
Llave: REGLEG

Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: RFC
Descriptor: FECALT

Nombre: MANEJADO_POR

Llave: NUMCLI
Llave: DIGITO
Llave: NUMAPO

Paso ternaria:

Nombre: PORTAFOLIOS

Llave: CLAPOR
Llave: NUMREG

Descriptor: VALUNI
Descriptor: FECALT
Descriptor: POSICI
Descriptor: TIPLIQ
Descriptor: INTCUS

Nombre: GP_FINANCIERO

Llave: CLAGRU

Descriptor: RFC
Descriptor: SECTOR
Descriptor: FECALT
Descriptor: ACCEMI
Descriptor: VALLIB

Nombre: FONDOS

Llave: NUMCLI

Descriptor: FIDUCI
Descriptor: REPRES

Nombre: SUCURSALES

Llave: NUMSUC

Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: RFC
Descriptor: CUSTOD
Descriptor: NUMSUC_1

Nombre: CLIENTES

Llave: NUMCLI

Descriptor: TIPCLI
Descriptor: PERJUR
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: VALCAR
Descriptor: EFECTI
Descriptor: NUMSUC

Nombre: NACIONAL

Llave: NUMCLI

Descriptor: COMISI
Descriptor: RFC
Descriptor: CLAFIS
Descriptor: MANCUE

Nombre: INTERNACIONAL

Llave: NUMCLI

Descriptor: TIPMON
Descriptor: NACION
Descriptor: CUSVAL
Descriptor: MANCUE
Descriptor: NUMGES
Descriptor: REGLEG

Nombre: PROMOTORES

Llave: DIGITO
Llave: NUMAPO

Descriptor: FECING
Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: REGLEG
Descriptor: RFC
Descriptor: NUMSUC
Descriptor: DIGITO_1
Descriptor: NUMAPO_1

Nombre: GESTORES

Llave: NUMGES
Llave: REGLEG

Descriptor: NOMBRE
Descriptor: DIRECC
Descriptor: CODPOS
Descriptor: RFC
Descriptor: FECALT

Nombre: MANEJADO_POR

Llave: NUMCLI
Llave: DIGITO
Llave: NUMAPO

Nombre: OPERA_EN

Llave: CLAPOR
Llave: NUMREG
Llave: CLAGRU
Llave: NUMCLI

ANEXO 4: RESULTADO DEFINITIVO DE LA TRADUCCION DEL SAPF

```
#name_rel
PORTAFOLIOS
#keys
CLAPOR
NUMREG
#endkeys
#desc
VALUNI
FECALT
POSICI
TIPLIQ
INTCUS
#enddesc
#name_rel
GP_FINANCIERO
#keys
CLAGRU
#endkeys
#desc
RFC
SECTOR
FECALT
ACCEMI
VALLIB
#enddesc
#name_rel
FONDOS
#keys
NUMCLI
#endkeys
#desc
FIDUCI
REPRES
#enddesc
#name_rel
SUCURSALES
#keys
NUMSUC
#endkeys
#desc
NOMBRE
DIRECC
CODPOS
RFC
CUSTOD
NUMSUC_1
#enddesc
#name_rel
CLIENTES
#keys
NUMCLI
#endkeys
```

#desc
TIPCLI
PERJUR
NOMBRE
DIRECC
CODPOS
VALCAR
EFECTI
NUMSUC
#enddesc
#name_rel
NACIONAL
#keys
NUMCLI
#endkeys
#desc
COMISI
RFC
CLAFIS
MANCUE
#enddesc
#name_rel
INTERNACIONAL
#keys
NUMCLI
#endkeys
#desc
TIPMON
NACION
CUSVAL
MANCUE
NUMGES
REGLEG
#enddesc
#name_rel
PROMOTORES
#keys
DIGITO
NUMAPO
#endkeys
#desc
FECING
NOMBRE
DIRECC
CODPOS
REGLEG
RFC
NUMSUC
DIGITO_1
NUMAPO_1
#enddesc
#name_rel
GESTORES
#keys

NUMGES
REGLEG
#endkeys
#desc
NOMBRE
DIRECC
CODPOS
RFC
FECALT
#enddesc
#name_rel
MANEJADO_POR
#keys
NUMCLI
DIGITO
NUMAPO
#endkeys
#desc
#enddesc
#name_rel
OPERA_EN
#keys
CLAPOR
NUMREG
CLAGRU
NUMCLI
#endkeys
#desc
#enddesc
#gene
NUMSUC_1
#fuente
NUMSUC
#gene
DIGITO_1
#fuente
DIGITO
#gene
NUMAPO_1
#fuente
NUMAPO

ANEXO 5: RESULTADO DE LA NORMALIZACION DEL SAPF

#name_rel
PORTAFOLIOS_a
#keys
CLAPOR
NUMREG
#endkeys
#desc
VALUNI
FECALT
POSICI
INTCUS
#enddesc
#name_rel
PORTAFOLIOS_b
#keys
INTCUS
#endkeys
#desc
TIPLIQ
#enddesc
#name_rel
GP_FINANCIERO
#keys
CLAGRU
#endkeys
#desc
RFC
SECTOR
FECALT
ACCEMI
VALLIB
#enddesc
#name_rel
FONDOS
#keys
NUMCLI
#endkeys
#desc
FIDUCI
REPRES
#enddesc
#name_rel
SUCURSALES
#keys
NUMSUC
#endkeys
#desc
NOMBRE
DIRECC
CODPOS
RFC
CUSTOD

NUMSUC_1
#enddesc
#name_rel
CLIENTES
#keys
NUMCLI
#endkeys
#desc
TIPCLI
PERJUR
NOMBRE
DIRECC
CODPOS
VALCAR
EFECTI
NUMSUC
#enddesc
#name_rel
NACIONAL
#keys
NUMCLI
#endkeys
#desc
COMISI
RFC
CLAFIS
MANCUE
#enddesc
#name_rel
INTERNACIONAL
#keys
NUMCLI
#endkeys
#desc
TIPMON
NACION
CUSVAL
MANCUE
NUMGES
REGLEG
#enddesc
#name_rel
PROMOTORES_a
#keys
DIGITO
NUMAPO
#endkeys
#desc
NOMBRE
DIRECC
CODPOS
NUMSUC
DIGITO_1
NUMAPO_1

#enddesc
#name_rel
PROMOTORES_b
#keys
NUMAPO
#endkeys
#desc
FECING
REGLEG
#enddesc
#name_rel
GESTORES
#keys
NUMGES
REGLEG
#endkeys
#desc
NOMBRE
DIRECC
CODPOS
RFC
FECALT
#enddesc
#name_rel
MANEJADO_POR
#keys
NUMCLI
DIGITO
NUMAPO
#endkeys
#desc
#enddesc
#name_rel
OPERA_EN
#keys
CLAPOR
NUMREG
CLAGRU
NUMCLI
#endkeys
#desc
#enddesc
#gene
NUMSUC_1
#fuente
NUMSUC
#gene
DIGITO_1
#fuente
DIGITO
#gene
NUMAPO_1
#fuente
NUMAPO

ANEXO 6: DISEÑO FISICO DE LA BASE DE DATOS DEL SAPF

```
#esquema SAPF
#diccionario
        CLAPOR,     A,      7,      "_______"
        VALUNI,     N,      10,     "__________"
        FECALT,     D,      6,      "__/__/__"
        POSICI,     N,      15,     "_______________"
        TIPLIQ,     A,      2,      "__"
        INTCUS,     A,      40,     "________________________________________"
        NUMREG,     Z,      7,      "_______"
        DIGITO,     A,      2,      "__"
        NUMAPO,     N,      10,     "__________"
        FECING,     D,      6,      "__/__/__"
        NOMBRE,     A,      60,     "____________________________________________________________"
        DIRECC,     A,      60,     "____________________________________________________________"
        CODPOS,     Z,      5,      "_____"
        REGLEG,     A,      40,     "________________________________________"
        RFC,        A,      14,     "______________"
        CLAGRU,     A,      7,      "_______"
        SECTOR,     A,      2,      "__"
        ACCEMI,     N,      15,     "_______________"
        VALLIB,     N,      10,     "__________"
        NUMCLI,     Z,      7,      "_______"
        TIPCLI,     A,      2,      "__"
        PERJUR,     A,      2,      "__"
        VALCAR,     C,      15,     "$_____.__"
        EFECTI,     C,      15,     "$_____.__"
        TIPMON,     A,      2,      "__"
        NACION,     A,      40,     "________________________________________"
        CUSVAL,     A,      40,     "________________________________________"
        MANCUE,     A,      2,      "__"
        COMISI,     N,      2,      "%__"
        CLAFIS,     A,      5,      "_____"
        FIDUCI,     A,      40,     "________________________________________"
        REPRES,     A,      60,     "____________________________________________________________"
        NUMGES,     Z,      5,      "_____"
        NUMSUC,     Z,      5,      "_____"
        CUSTOD,     C,      15,     "$_____.__"
#fin diccionario

#archivo PORTAFOLIOS_a
        CLAPOR
        NUMREG
        VALUNI
        FECALT
        POSICI
        INTCUS
#fin archivo

#archivo PORTAFOLIOS_b
        INTCUS
```

```
        TIPLIQ
#fin archivo

#archivo GP_FINANCIERO
        CLAGRU
        RFC
        SECTOR
        FECALT
        ACCEMI
        VALLIB
#fin archivo

#archivo FONDOS
        NUMCLI
        FIDUCI
        REPRES
#fin archivo

#archivo SUCURSALES
        NUMSUC
        NOMBRE
        DIRECC
        CODPOS
        RFC
        CUSTOD
        NUMSUC_1
#fin archivo

#archivo CLIENTES
        NUMCLI
        TIPCLI
        PERJUR
        NOMBRE
        DIRECC
        CODPOS
        VALCAR
        EFECTI
        NUMSUC
#fin archivo

#archivo NACIONAL
        NUMCLI
        COMISI
        RFC
        CLAFIS
        MANCUE
#fin archivo

#archivo INTERNACIONAL
        NUMCLI
        TIPMON
        NACION
        CUSVAL
        MANCUE
```

```
        NUMGES
        REGLEG
#fin archivo

#archivo PROMOTORES_a
        DIGITO
        NUMAPO
        NOMBRE
        DIRECC
        CODPOS
        NUMSUC
        DIGITO_1
        NUMAPO_1
#fin archivo

#archivo PROMOTORES_b
        NUMAPO
        FECING
        REGLEG
#fin archivo

#archivo GESTORES
        NUMGES
        REGLEG
        NOMBRE
        DIRECC
        CODPOS
        RFC
        FECALT
#fin archivo

#archivo MANEJADO_POR
        NUMCLI
        DIGITO
        NUMAPO
#fin archivo

#archivo OPERA_EN
        CLAPOR
        NUMREG
        CLAGRU
        NUMCLI
#fin archivo


#llave  PORTAFOLIOS_a  CLAPOR, NUMREG
#llave  PORTAFOLIOS_b  INTCUS
#llave  GP_FINANCIERO  CLAGRU
#llave  FONDOS         NUMCLI
#llave  SUCURSALES     NUMSUC
#llave  CLIENTES       NUMCLI
#llave  NACIONAL       NUMCLI
#llave  INTERNACIONAL  NUMCLI
#llave  PROMOTORES_a   DIGITO, NUMAPO
```

```
#llave PROMOTORES_b  NUMAPO
#llave GESTORES      NUMGES, REGLEG
#llave MANEJADO_POR  NUMCLI, DIGITO, NUMAPO
#llave OPERA_EN      CLAPOR, NUMREG, CLAGRU, NUMCLI

#alias NUMSUC_1      NUMSUC
#alias DIGITO_1      DIGITO
#alias NUMAPO_1      NUMAPO

#fin esquema SAPF
```

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL I.P.N.

Avenida Instituto Politécnico Nacional 2508 Col. San Pedro Zacatenco
México, D.F. C.P. 07300

EL JURADO DESIGNADO POR LA SECCION DE COMPUTACION DEL DEPARTAMENTO DE INGENIERIA ELECTRICA, APROBO EL DIA 23 DEL MES DE MARZO DEL AÑO DE 1994 TRABAJO DE TESIS.

"SISTEMAS PARA DISEÑO DE BASES DE DATOS EVE"

DESARROLLADO POR EL ALUMNO:

RAUL HERNANDEZ STEFANONI

Dr. Sergio V. Chapa Vergara

José Oscar Olmedo Aguirre

Guillermo Rafael Domínguez de León

Tel.: Conmutador 754-02-00 *Telex: 1772826* *Telefax: 586-65-64*

AUTOR HERNANDEZ STEFANONI, R.		
TITULO SISTEMA PARA DISEÑO DE BASES DE DATOS EVE		
CLASIF. XM 94.2 **RGTRO** BI 14010		
NOMBRE DEL LECTOR	FECHA PREST.	FECHA DEVOL.
NOE SIERRA ROMERO	4/1/95	26/1/95
Noe Sierra Romero	31/1/95	21 feb 95
JESUS NARO CRUZ	1/3/95	22-III-95
Carlos A.H.H.	5/4/95	7-IV-95
NOE SIERRA ROMERO	11/9/95	5/10/95
Héctor Díaz Rojas	1/11/95	10/10/95
Dr. Guillermo Morales	13-12-95	13 DIC/95
[illegible]	7-12-95	
SERENO PEDALOZA	16/01/	
MARCO RUEDA	2	
FCO. SOUZ		
Martha R. López		