ESCUELA SUPERIOR
POLITECNICA DEL LITORAL

ESCUELA DE CIENCIAS DE LA COMPUTACION

BASE DE DATOS RELACIONAL
POLIREL

VERSION 2.0

Presentada por:

ANA COX VASCONEZ

VERONICA CUCALON ZENCK

AGUSTIN CHUSAN VITERI

FAUSTO JACOME LOPEZ

FERNANDO PAEZ GALARZA

MATILDE PERE YCAZA

GISSELLE SALMON BOWEN

Director:

ING. ANDRES NAVARRO GAZZOLA

Guayaquil-Ecuador

1984

DECLARACION EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestos en esta tesis, nos corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL".

(Reglamento de Examenes y Titulos profesionales de la ESPOL).

ANA COX VASCONEZ	VERONICA CUCALON ZENCK
AGUSTIN CHUSAN VITERI	FAUSTO JACOME LOPEZ
FERNANDO PAEZ GALARZA	MATILDE PERE YCAZA

GISSELLE SALMON BOWEN

MANUAL DEL SISTEMA

PROLOGO A LA VERSION 2.0

La nueva version de Polirel es producto del desarrollo y madurez evolutivos del propio sistema, en union con el esfuerzo del grupo de estudiantes que hicieron posible este trabajo.

Polirel en esta version ha sido complementado y mejorado notablemente, no solo en la capacidad del sistema en si, sino tambien en detalles llenos de sutilezas en pro de una mejor aceptacion del producto, manteniendo su ideologia basica y fundamental de ser un Software en principio educacional.

El objetivo primordial de esta version consistio en proporcionar una gran gama de utilitarias que se traduzcan en una mayor flexibilidad en el uso del sistema por parte del usuario, y que ademas, sirva como material de ensenanza integral en la tecnologia de Base de Datos, considerando aspectos de diseno, optimizacion y sintonizacion de las aplicaciones.

Las Utilitarias de Polirel abarcan un gran cantidad de conceptos y propositos practicos, tales como generacion de indices invertidos para optimizacion de las consultas del Query, generacion de indices clasificados para efectos de generacion de reportes, formateo de relaciones, carga y descarga de relaciones, consultas catalogadas, archivo de

auditoria, entre otros.

Como modulo aparte el generador de reportes de Polirel, permite en forma sencilla y versatil la creacion, modificacion de archivos de formatos de reportes para manipulacion de los datos almacenados en Polirel.

Otro de los logros de esta nueva version, es la capacidad operativa de Polirel a traves de un programa aplicacional cualquiera escrito en su lenguaje anfitrion PASCAL. Un usuario puede leer y actualizar cualquier archivo de la Base de Datos utilizando un programa PASCAL y el modulo de interface de Polirel.

La capacidad de almacenamiento de las tuplas ha sido expandida y podra ser seleccionada por el usuario segun requerimientos de cada aplicacion especifica. De igual forma, la implementacion de numeros reales y gran mejoramiento de la presentacion del sistema total, hacen de Polirel un sistema de Base de Datos educativo de gran contenido didactico, asi como de elevado nivel tecnico.

PROLOGO A LA VERSION 1.0

La idea de este proyecto se origino ante la necesidad de materializar la teoria y conceptos relacionados con la tecnologia de Base de Datos, y de manera especifica el modelo relacional.

El dictado de este topico durante varios semestres en la catedra de Sistemas de Computo, motivo a la consecucion de un producto terminado, que para propositos practicos y netamente didacticos fue evolucionando hasta convertirse en una realidad satisfactoria.

La intencion al concebir este proyecto fue desde un principio bien clara. No se trataba de desarrollar un producto comercial, sino un sistema que permita al estudiante trabajar y afianzar sus conocimientos que sobre el tema haya sido presentado durante el semestre.

Sin embargo, la concepcion del mismo ha sido efectuada en forma modular y sistematica que bien pudiera servir para un proyecto de investigacion mas ambicioso.

El desarrollo de este proyecto abarca toda una gama de conceptos sobre el area de computacion, no solamente sobre Base de Datos Relacional que es el tema de fondo, sino tambien estructuras de datos, desarrollo de una gramatica sencilla y un procesador para el Query final. Todo esto, teniendo en mente como principal objetivo al usuario, que a

traves de un lenguaje algebraico relacional pudiera en forma interactiva hacer uso en forma sencilla de los diferentes modulos del sistema.

El Sistema consta de una leccion por pantalla, la cual tiene como proposito orientar al usuario en la parte netamente operativa del sistema en conjunto, asi como presentar la teoria pertinente a los fundamentos del Modelo Relacional y el lenguaje algebraico. Otro modulo del Sistema esta constituido por el Editor de Tuplas, el cual permite el ingreso, eliminacion, modificacion y consulta de cualquier tupla en cualquiera de las relaciones creadas por el usuario.

Finalmente, el modulo del Query que a manera de interpretador dinamico permite al usuario formular los diferentes procedimientos y consultas al Sistema Relacional.

NOTA: La direccion de este proyecto, deja por medio de la presente nota constancia del reconocimiento al trabajo de los siguientes alumnos que hicieron posible la Version 1.0: Mirza Arteaga, Xavier Cardenas, Julio Chang, Reinaldo Roca y Maria Mercedes Villacreses.

I N D I C E

1. INTRODUCCION

POLIREL es un sistema de Base de datos seudo relacional, cuyo proposito es netamente academico. El sistema ha sido disenado bajo caracteristicas de vistas estaticas constituidas por las relaciones especificadas por el usuario y estructuras relacionales vigentes durante una sesion de trabajo. Es decir, todas las estructuras permanentes/temporales realizadas mediante interaccion con el pre-procesador del QUERY son validas durante la sesion de trabajo solamente.

Cualquier JOIN, SELECCION, PROYECCION o LIST/DISP es ejecutado por el QUERY en memoria; el sistema no salva estas estructuras permanentemente al final de la sesion. Sin embargo, el usuario mantendra en forma definitiva todas aquellas relaciones por el creadas, pudiendo ser las mismas modificadas mediante el Editor de tuplas o a traves de un programa PASCAL, utilizando la Unidad Polirel, residente en la Libreria del Sistema; y a partir de las cuales cualquier estructura podra ser generada en una nueva sesion.

De igual manera, las consultas realizadas dentro del QUERY pueden ser mantenidas en un archivo generado a traves del modulo CONSULTAS CATALOGADAS dentro del sistema UTILITARIAS, lo que le permitira ejecutar dichas consultas sin necesidad de volverlas a digitar.

Otra herramienta a disposicion del usuario es el GENERADOR DE REPORTES, el cual facilita la obtencion y presentacion de los datos.

2. GENERALIDADES

Todo el Sistema de la Base de Datos POLIREL esta dividido en cinco modulos:

1) Leccion de uso de la Base
2) Editor de Tuplas
3) Query o Interpretador
4) Utilitarias del Sistema
5) Generador de Reportes

Cada uno de los modulos fue desarrollado en forma independiente. El Sistema fue realizado en su totalidad en un computador APPLE II PLUS, utilizando PASCAL como lenguaje de implementacion.

3. ORGANIZACION FISICA DE LA BASE

3.1 RELACIONES

Los nombres de relaciones estan constituidos por una letra desde la 'A' hasta la 'Z'. Dentro de la Base de Datos Polirel, cada Relacion consta de 3 archivos:

- Archivo Descriptor de Tuplas (ADT)
- Archivo de Tuplas (AT)
- Archivo de Datos Alfanumericos (ADA)

Estos archivos son creados fisicamente con la opcion FORMATEO DE ARCHIVOS del modulo UTILITARIAS. (Ver Numeral 9.1)

El nombre con que son creados estos archivos esta formado por: los caracteres '#5:', que indican que los archivos estaran en el DRIVE 2; el nombre de la relacion como identificacion, luego un numero que sera:

1 --> Archivo descriptor de Tuplas

2 --> Archivo de datos Alfanumericos

3 --> Archivo de Tuplas

y por ultimo los caracteres '.DATA', que especifican que se trata de un archivo de datos.

Estos 3 archivos estaran formados por uno o varios bloques, que seran tratados por todos los programas de la Base como arreglos empaquetados de 512 bytes cada bloque.

Pudiendo el Archivo de Tuplas asi como el Archivo de

Datos Alfanumericos contener mas de un bloque, segun requerimientos del usuario.

3.1.1 ARCHIVO DESCRIPTOR DE TUPLAS (ADT)

El archivo descriptor de tuplas (ADT) sirve para describir las columnas que tiene la relacion y proporciona informacion general sobre la misma. La descripcion de las columnas es la siguiente:

POSICION	DESCRIPCION
1-1	Estrategia de la columna (E=entero, C=caracter, R=real, S=string)
2-4	Posicion donde comienza el primer dato de la columna en el archivo AT
5-5	Numero del bloque donde comienza el nombre de la columna en el archivo ADA
6-8	Posicion donde comienza el nombre de la columna en el archivo ADA
9-10	Longitud del nombre de la columna en el archivo ADA
11-11	Marca de eliminacion de la columna (*=eliminada, BLANCO=no eliminada)

La descripcion anterior corresponde a la primera columna de una relacion, en el caso de que existan mas columnas las siguientes 11 posiciones corresponderian a la segunda columna, las proximas

11 posiciones a la tercera y asi sucesivamente. Cabe indicar que se ha restringido el numero de columnas que puede tener una Relacion a 10.
La descripcion sobre la informacion general de la Relacion es la siguiente:

POSICION	DESCRIPCION
300-329	Numero de tuplas habiles para cada bloque
350-359	Marca de existencia de archivos de clasificacion (X = si existe archivo clasificado BLANCO = no existe)
400-407	Numeros de columnas sobre las que se ha generado archivo de Indices Invertidos Maximo 4 indices. (99=numero de columna para la que si existe indice BLANCO=no existe indice)
408-408	Estado de la Relacion ('*' = Relacion eliminada, 'F' = Relacion formateada, BLANCO = Relacion no esta eliminada)

409-410	Numero de tuplas que tiene la Relacion
411-412	Numero de columnas que tiene la Relacion
413-415	Longitud de la tupla en el archivo AT
416-416	Numero del bloque del archivo ADA que tiene la siguiente posicion disponible.
417-419	Siguiente posicion disponible en el archivo ADA
420-420	Numero de bloques que forman el archivo AT (0..9)
421-421	Numero de bloques que forman el archivo ADA (0..9)

3.1.2 ARCHIVO DE TUPLAS (AT)

El archivo de Tuplas (AT) tiene los datos en si de las tuplas de la relacion.

En el archivo de tuplas (AT) la longitud de cada campo esta dada por la estrategia de la columna, de acuerdo a la siguiente tabla:

ESTRATEGIA	LONG. CAMPO
C	1 byte
E	5 bytes
R	7 bytes
S	7 bytes

Cabe indicar que para el caso de cadenas de caracteres (S), las 7 posiciones se dividen de la siguiente manera:

COLUMNA	DESCRIPCION
1-1	Numero del bloque en el archivo ADA donde esta el dato
2-4	Posicion donde comienza el dato en el archivo ADA
5-7	Longitud del dato en el archivo ADA

Para la estrategia entero (E), el dato sera un

numero ajustado a la derecha; para la estrategia real (R) las cinco primeras posiciones del dato corresponderan a la parte entera del numero y las dos restantes a la parte decimal; y para la estrategia caracter (C) el dato sera un caracter alfabetico o numerico.

La longitud de la tupla en el archivo AT, estara dada por la suma de las longitudes de las estrategias de las columnas mas una posicion usada como marca de eliminacion. (*=tupla eliminada, BLANCO=no eliminada)

3.1.3 ARCHIVO DE DATOS ALFANUMERICOS (ADA)

El Archivo de Datos Alfanumericos dependiendo de los requerimientos del usuario puede tener mas de un bloque, contiene los datos correspondientes a los nombres de la columnas y a aquellas columnas cuya estrategia sea string (S); para mayor facilidad en el manejo de cadenas de caracteres, estas no son compartidas entre bloques.

3.2 OTROS ARCHIVOS

3.2.1 ARCHIVO DE INDICES INVERTIDOS

Este archivo es generado cada vez que se ejecuta la opcion de Indices Invertidos del modulo Utilitarias. El nombre con que se genera el archivo tiene el siguiente formato:

#5:rC99.DATA

Donde:

r, es el nombre que identifica la relacion;

C, es una constante que diferencia el archivo de los otros utilizados.

99, es el numero de la columna sobre la que se ha generado el indice (01-10);

.DATA, es una constante que indica que se trata de un archivo de datos.

Tiene dos tipos de bloque:

1. Bloque Directorio.-

POSICIONES	DESCRIPCION
2	Total de buckets
1 5 7 10	Valor del Atributo
2	Numero de bucket asociado

Las dos primeras posiciones corresponden siempre al numero total de buckets que tiene el archivo, es decir que a partir de el, se puede obtener el numero de bloques que lo forman.

Comenzando desde la posicion 3 se grabaran los diferentes valores del atributo, cada uno asociado a un numero de bucket, en el que se encontraran los numeros relativos de las tuplas que contengan el valor referenciado.

La longitud del valor del atributo esta dada por su estrategia:

ESTRATEGIA	LONGITUD
'C'	1
'E'	5
'R'	7
'S' (*)	10

Por lo tanto el numero de posibles entradas en el bloque directorio dependera de la estrategia del atributo.

(*) Para generar un indice sobre un atributo cuya estrategia es string; se consideraran solo los 10 primeros caracteres.

2. Bloques de Buckets.-

Todos los demas bloques del archivo seran

divididos en buckets; cada bloque esta formado por ocho buckets distribuidos de la siguiente manera:

POSICION	DESCRIPCION
1-64	Bucket 1
65-128	Bucket 2
129-192	Bucket 3
193-256	Bucket 4
257-320	Bucket 5
321-384	Bucket 6
385-448	Bucket 7
449-512	Bucket 8

Y cada bucket contendra:

POSICION	DESCRIPCION
1-2	Siguiente bucket encadenado a el
3-4	Siguiente posicion disponible dentro del bucket
5-7	Numero relativo de tupla
8-10	Numero relativo de tupla

.
.
.

alcanzando hasta 20 entradas por bucket.

Si las posiciones 1-2 se encuentran en blanco significa que no tiene ningun bucket encadenado.

Si la siguiente posicion disponible dentro del bucket es igual a 'O' representara que el bucket esta lleno y probablemente tendra otro bucket encadenado a el.

3.2.2 ARCHIVO DE CLASIFICACION

El archivo de clasificacion tiene los numeros relativos de las tuplas distribuidos en un solo bloque de la siguiente manera:

POSICION	No.BUCKET
1- 64	1
65-128	2
129-192	3
193-256	4
257-320	5
321-384	6
385-448	7
449-512	8

Cada bucket contendra:

POSICION	CONTENIDO
1- 2	Siguiente bucket encadenado a el
3- 4	Siguiente posicion disponible dentro del bucket
5- 7	Numero relativo de tupla
8-10	Numero relativo de tupla
.	
.	

alcanzando hasta 20 entradas por bucket.

Si las posiciones 1-2 se encuentran en blanco significa que no tiene ningun bucket encadenado.

Si la siguiente posicion disponible dentro del bucket es igual a '00' significa que el bucket esta lleno y probablemente tendra otro bucket encadenado a el.

El nombre del archivo de clasificacion esta constituido por la constante '#5:' seguido del nombre de la relacion, luego la constante 'S', numero de la columna considerada como llave primaria (2 digitos) y por ultimo la constante '.DATA'.

3.2.3 ARCHIVO DE REORGANIZACION

El nombre con que se crea este archivo esta constituido por los caracteres '#5:', que indica que el archivo estara en el DRIVE 2; a continuacion el caracter 'D' para diferenciarlo de otros archivos que utiliza la Base de Datos en su organizacion fisica; luego el nombre de la relacion seguido por los caracteres '.DATA'.

El archivo de reorganizacion tiene dos tipos de bloques; un bloque directorio que contiene informacion similar al archivo ADT distribuida de la siguiente manera:

POSICION	DESCRIPCION
1-1	Estrategia de la columna
2-4	Posicion donde comienza el primer dato de la columna en el archivo AT
5-5	Numero del bloque donde comienza el nombre de la columna en el archivo ADA
6-8	Posicion donde comienza el nombre de la columna en el archivo ADA
9-10	Longitud del nombre de la columna en el archivo ADA

11-11	Marca de eliminacion de la columna

y ademas, la siguiente informacion general:

POSICION	DESCRIPCION
408-408	Estado de la relacion
409-410	Numero de tuplas que tiene la relacion
411-412	Numero de columnas que tiene la relacion
413-415	Longitud de la tupla en el archivo AT
416-416	Numero del bloque del archivo ADA que tiene la siguiente posicion disponible
417-419	Siguiente posicion disponible en el archivo ADA
420-420	Numero de bloques que forman el archivo AT
421-421	Numero de bloques que forman el archivo ADA.

Tambien tiene otro tipo de bloque que sirve para almacenar los nombres de las columnas y los datos para cada una de las tuplas, con la particularidad que utiliza el separador '&' para cada dato.

3.2.4 ARCHIVO DE AUDITORIA

El nombre con que se identifica el Archivo de Auditoria esta formado por los caracteres '#5:' para indicar que el archivo se encuentra en el DRIVE 2; luego, los caracteres 'AUDIT' para diferenciarlo de otros archivos y a continuacion la constante '.DATA'.

En este archivo se grabaran las expresiones que se realicen bajo una sesion de trabajo en el Query. Estos enunciados o expresiones ademas estaran separados por un '&' y su contenido se encuentra encriptado.

3.2.5 ARCHIVO DE CONSULTAS CATALOGADAS

El archivo de consultas catalogadas tiene en un bloque, la marca de eliminacion de la linea y la expresion, para las 10 lineas permitidas en las consultas catalogadas, de la siguiente manera:

POSICION	CONTENIDO
1-1	Marca de eliminacion
2-50	Expresion 1
51-51	Marca de eliminacion
52-100	Expresion 2
.. ...	.....
.. ...	.....
451-451	Marca de eliminacion
452-500	Expresion 10

El nombre del archivo de consultas catalogadas esta constituido por la constante '#5:C', seguido del nombre de la consulta catalogada que puede ser de hasta 2 caracteres, debiendo el primero ser siempre una letra y por ultimo seguido de la constante '.DATA'.

4. ACCESO A POLIREL A TRAVES DE UN PROGRAMA PASCAL

4.1 INTRODUCCION

En la primera version de la Base de Datos Polirel el acceso a las relaciones (archivos) estaba reservados unicamente para los modulos de la base como el Editor y el Query, por lo tanto cualquier proceso externo con dichas relaciones no era posible; bajo esta premisa surgio la necesidad de implementar un modulo que permita a cualquier programador en pascal accesar y manipular estas relaciones tan facilmente como cualquier archivo pascal.

Un modulo que permita realizar lo antes descrito no podia ser facilmente implementado utilizando el pascal tradicional, lo que nos llevo a utilizar el pascal desarrollado en la Universidad de California (UCSD Pascal), el cual permite entre otras cosas que el programa principal sea realizado en segmentos lo que fue muy util en la implementacion de un programa grande y complejo como es la Base de Datos "Polirel" (version 2.0).

La caracteristica mas importante y en la cual se basa este modulo es el concepto de unidad (unit); un grupo de uno o mas procedimientos y/o funciones a nivel de lenguaje fuente pueden ser compilados separadamente como unidad, el resultado de compilar una unidad es un

archivo de libreria, el cual puede ser usado directamente o puede ser incorporado en la libreria del sistema como es el caso de la Unidad Polirel.

La unidad Polirel ha sido desarrollada de tal manera que facilite la tarea del programador en pascal; cualquier programa puede invocar los procedimientos de la Unidad sin preocuparse en lo mas minimo en el proceso de enlace (linkedicion) ya que este es realizado automaticamente cuando el programa del usuario inicia su ejecucion, ademas la unidad Polirel dispone de ciertos tipos de datos, variables, funciones y procedimientos que facilitan la programacion.

Esperamos que la unidad "Polirel" sea para los usuarios de la base de datos Relacional y especialmente para programadores una herramienta imprescindible en sus tareas y para aquellos programadores novatos sea materia de estudio y experimentacion.

4.2 LA UNIDAD "POLIREL"

El texto fuente de la unidad Polirel tiene la forma de un programa pascal, el cual ha sido incorporado en la libreria del sistema siendo un modulo permanente en esta; declarada como una unidad intrinsica, ocupa el segmento 23 (code) y segmento 24 (data) de la libreria del sistema. Polirel no es ejecutable en lugar de ello puede ser utilizaddo por uno o varios programas.

Una libreria del sistema asi implementada contiene el codigo de funciones y procedimientos los cuales son disponibles para cualquier programa que usa la libreria; para utilizarlos el programa necesita tener la siguiente declaracion:

```
USES POLIREL;
```

despues del encabezado del programa; de esta forma el programa podra utilizar todos aquellos procedimientos que se describiran mas adelante.

La implementacion de la libreria del sistema utilizada en el desarrollo de la Base de Datos Relacional involucra varios pasos:

- Escribir la unidad Polirel en un lenguaje fuente (pascal) como cualquier programa pero apegado a las reglas sintacticas para el desarrollo de una unidad.

- Compilacion de la unidad cuyo proceso es similar a compilar cualquier programa pascal y cuyo resultado es un archivo de libreria.
- Insertar este archivo de libreria obtenido dentro de la libreria del sistema, utilizando la utilitaria LIBRARY del sistema operativo Pascal.

De esta manera la unidad Polirel ha sido instalada en la libreria del sistema, es importante en esta parte que el usuario de Polirel especialmente programadores revisen la lista de palabras reservadas en el apendice C, ya que el interface de la unidad contiene ciertas variables, funciones y procedimientos utilizados en la implementacion de la base de Datos relacional Polirel ademas de los procedimientos descritos en este manual y que estan disponibles para el usuario.

4.2.1 ESTRUCTURA:

La estructura de la unidad Polirel es la siguiente:

- Un encabezado que lo identifica;

```
UNIT POLIREL
intrinsic code 23
data 24;
```

- Un interface :

  La cual define la forma en que el programa huesped se comunica con los procedimientos y funciones de la unidad, aqui han sido definidos aquellas constantes, tipos, variables, funciones y procedimientos que son publicos y pueden ser accesados por el programa huesped como si estos hubiesen sido definidos en el.

  Para una completa informacion de todos aquellos types, variables, funciones y procedimientos disponibles en la libreria del sistema de la Base el usuario puede obtener un mapa de la libreria con la utilitaria LIBMAP ver Apple II Operating System Reference Manual.

- Una implementacion:

  En la cual se describen las funciones y procedimientos por si mismos; y,

- Una inicializacion:

Que es el programa principal de la unidad el cual se ejecutara automaticamente cuando el programa huesped inicie su ejecucion. Aqui se inicializan todas aquellas variables y estructuras utilizadas por la base de datos y la unidad Polirel.

4.2.2 OPERACION:

La unidad Polirel se comunica con los programas que la invocan por medio de la interface,la cual sirve de referencia para que el usuario defina el formato de su registro de trabajo, y la unidad a su vez basado en este formato retornara los correspondientes datos.

El usuario por lo tanto esta obligado a definir un formato de registro (Ver Manual Operativo, Numeral 8.5),para lo cual se ha definido lo siguiente:

```
BLOCK = PACKED ARRAY [0..10] OF STRING[7];
```

En cada ocurrencia, el primer caracter del string representara la estrategia de la columna y los seis restantes seran ocupados por su nombre; varias columnas descritas en esta forma representan un registro de trabajo.

Como la Base de Datos maneja cuatro estrategias (e,c,r,s) y cada una de ellas representa su correspondiente tipo de datos ,en la unidad se ha definido:

```
ENPOL: PACKED ARRAY [0..10] OF INTEGER;
CAPOL: PACKED ARRAY [0..10] OF CHAR;
STPOL: PACKED ARRAY [0..10] OF STRING;
REPOL: PACKED ARRAY [0..10] OF REAL;
```

en donde se almacenaran los datos solicitados por

el usuario dependiendo de su estrategia.

La mayoria de los procedimientos residentes en la unidad Polirel tienen una variable entera global la cual tomara determinados valores para indicar el resultado de la operacion realizada; esta variable se la ha definido como:

```
OPCONDICION : INTEGER;
```

El usuario puede examinar el valor de esta variable para determinar si la operacion realizada ha sido exitosa, el significado de todos los posibles valores que puede tomar esta variable se da en el Apendice B. Es responsabilidad absoluta del programador verificar si la operacion realizada ha sido exitosa o no, asi se evitara resultados inesperados.

4.2.3 EL SISTEMA MULTI-BLOQUE:

La forma como la unidad polirel maneja las relaciones que pueden tener varios bloques y como influyen estos en los procedimientos de actualizacion,eliminacion,insercion y lectura de tuplas sera descrito a continuacion.

Un punto critico al manipular un archivo multibloque es determinar exactamente el numero del bloque en el que se encuentra una tupla dada y la posicion fisica de dicha tupla dentro del bloque.

Para determinar el bloque en el que se encuentra una tupla se ha definido el siguiente arreglo:

```
TUPLALOGICA: PACKED ARRAY [0..10] OF INTEGER;
```

Cada ocurencia del arreglo indicara el numero del bloque al que se refiere y su contenido indicara el numero de tuplas activas dentro de cada bloque, la posicion fisica de una tupla dentro del bloque se lo encuentra rastreando todo el bloque (pasando por alto las tuplas eliminadas) hasta alcanzar la tupla requerida.

Este arreglo sera debidamente actualizado en los procedimientos que modifiquen el numero de tuplas activas existentes.

Ademas la unidad polirel contiene variables

enteras globales (uno por cada archivo de la relacion) los cuales indicaran el numero de bloque que se encuentra en memoria en un momento dado, solo la falta de un bloque en memoria ocasionara que este sea leido del disco, pero antes de leer el nuevo bloque se verificara si los datos residentes en los bloques de trabajo (memoria) no han sido modificados, es decir, que los procedimientos de actualizacion , eliminacion e insersion de tuplas no hayan sido activados en cuyo caso debera primero protegerse los datos de memoria.

Todos aquellos procesos que ocasionen cambios en los datos residentes en memoria como los procedimientos ACTUALTU, ELIMINTU, ADICIOTU, deben finalizar con una sentencia FINPROCESO la cual permitira a la unidad Polirel mantener la integridad de los datos contenidos en la Base de Datos.

4.2.4 SISTEMA MULTI-ARCHIVO

El sistema multi-archivo implementado en la unidad polirel permite procesar concurrentemente hasta tres archivos (esto incluye archivos invertidos,normales y de clasificacion), Esto hace posible que el usuario lea alternadamente hasta tres relaciones. Las configuraciones de lectura permitidas por la unidad polirel es una de las siguientes:

- 1 a 3 relaciones normales;
- 1 a 2 relaciones normales y un archivo invertido
- 1 a 2 relaciones normales y un archivo de clasificacion.
- solo un archivo invertido
- solo un archivo de clasificacion.

Cuando un programa invoca un procedimiento de lectura, el sistema multi-archivo es activado, el cual mantiene en memoria el nombre de los archivos que estan siendo procesados, para lo cual se ha definido:

```
MFILES: PACKED ARRAY [0..4] OF CHAR
```

Cada ocurrencia representa una entrada en el sistema multi-archivo, las ocurrencias 0 y 4 son

utilizadas por la unidad para el control de los archivos.

Inicialmente no existe ningun archivo en el sistema, por lo tanto para todos aquellos archivos que son invocados por primera vez existe una fase de carga del archivo, esta fase incluye:

- Validacion del nombre
- Apertura de los archivos
- Inicializacion de los apuntadores correspondientes
- Lectura del respectivo bloque del archivo descriptor de tuplas (ADT), del cual se toman los parametros de trabajo para esa relacion tales como: numero de tuplas (NUMTUPLAS), numero de columnas (NUMCOLUMNAS), longitud de la tupla (LONGTUPLA), etc.
- Inicializacion de booleanas para condicionar ciertos procesos como la lectura de los bloques de los archivos AT y ADA.

Para aquellos archivos que ya tienen una entrada en el sistema multi-archivo se cumplen dos etapas: la primera que es la etapa de proteccion de los bloques de trabajo con todos sus respectivos parametros y segundo la etapa de restauracion de los bloques del archivo solicitado.

La etapa de proteccion incluye:

- Determinar exactamente cual es la relacion que se va a proteger para direccionar a los bloques reservados para tal motivo;
- Mover la informacion contenida en los bloques de trabajo a los bloques reservados para dicha relacion, ademas se protege todas las variables y parametros actuales de dicha relacion.

La etapa de restauracion involucra un proceso inverso al de la etapa de proteccion, esto es:

- Determinar exactamente cual es la relacion que se restaura para asi poder direccionar a los bloques correspondientes para la recuperacion de los datos;
- Cargar los bloques de trabajo con los datos contenidos en los bloques reservados para dicha relacion;
- Restauracion de ciertos parametros que fueron protegidos y la regeneracion de otros.

En cada una de estas fases se tiene el cuidado de actualizar correctamente los campos que indican cual es el archivo corriente de trabajo y los cambios realizados en el sistema multi-archivo.

De esta manera se mantiene en memoria los datos de la relacion que se desea procesar, por lo tanto

los procedimientos de lectura se limitan a procesar los datos dejados en los bloques de trabajo.

4.2.5 OPERACIONES PERMITIDAS POR LA UNIDAD POLIREL

- Lectura secuencial de una relacion, se leera tupla por tupla en el orden en que fueron creadas considerando solamente las tuplas no eliminadas;
- Lectura directa de una tupla para lo cual la clave es la posicion fisica de la tupla dentro de la relacion;
- Lectura secuencial bajo un criterio de seleccion de todas aquellas tuplas que cumplen con dicho criterio (proceso de archivos invertidos);
- Lectura secuencial de un archivo de clasificacion con la finalidad de que el usuario procese sus relaciones en un orden determinado.
- Actualizacion de la tupla previamente leida.
- Eliminacion de la tupla previamente leida.
- Adicion de tuplas a las ya existentes dentro de una relacion dada.

Los procedimientos que nos permiten realizar todas estas funciones seran descritos a continuacion.

4.3 PROCESO SECUENCIAL DE RELACIONES

4.3.1 LECTURA SECUENCIAL DE UNA TUPLA

Procedimiento: LECSECTU.

OBJETIVO:

Un archivo que ha sido creado por el editor de la base de datos puede ser leido secuencialmente tupla a tupla y en el orden en el que fueron creadas gracias al modulo de lectura secuencial LECSECTU implementado en la libreria del sistema, este procedimiento procesara solamente las tuplas habiles, es decir, no eliminadas.

FORMATO:

LECSECTU(relacion,tabla-de-atributos)

donde:

Relacion:especifica el nombre de la relacion a ser accesada, el nombre debe ser un caracter alfabetico y debe haber sido creada por el editor de la base.

Tabla-de-atributos: especifica el formato de registro definido por el usuario para la relacion presente (ver definicion del formato de registro en el Manual Operativo de la Base), este formato

de registro sera constante para cada proceso, es decir, no podra ser alterado durante el proceso de la relacion.

DESCRIPCION:

La invocacion por parte de un programa al modulo LECSECTU hace que se active el sistema multi-archivo el cual controlara los archivos de proceso como se describio anteriormente.

Una vez que el sistema multi-archivo le indica que la operacion de carga/restauracion de la relacion ha sido exitosa se procede a validar si no se ha llegado al final de datos, es decir que el numero de tuplas leidas no sea mayor al numero de tuplas existentes dentro de la relacion, en cuyo caso se emitira el OPCONDICION correspondiente a dicha situacion, ademas se encendera una booleana (FINDATA=TRUE) para indicar el fin de datos; tanto la booleana como el valor de la variable OPCONDICION pueder ser utilizados por programadores para condicionar sus procesos.

Si no se ha detectado el fin de datos se pasa a validar el formato de registro definido por el usuario para lo cual se procede a obtener el nombre y la estrategia de todas las columnas de la

relacion y se las compara con las descritas por el usuario en la definicion de su registro, en caso de existir incoherencias se cancelara el procedimiento LECSECTU emitiendo el respectivo valor en OPCONDICION.

La recuperacion de datos se realiza columna a columna para lo que se ha implementado un arreglo que contiene la posicion fisica de cada columna dentro de la relacion; como el proceso de validacion del registro definido por el usuario se lo realiza solo la primera vez las subsecuentes lecturas solo referenciaran a este arreglo y procederan a retirar los datos respectivos.

Como todos los datos de la base se graban como cadena de caracteres, en la unidad Polirel existen las correspondientes rutinas de conversion de datos que permiten darle al usuario el correcto tipo de datos que solicitare.

4.3.2 LECTURA DIRECTA DE UNA TUPLA

En cada invocacion al modulo LECSECTU automaticamente se incrementara en uno el contador de tuplas leidas (TUPLANUMERO) hasta detectar el fin de datos, como este contador es una variable entera global su valor puede ser alterado por el programador, esta operacion resulta un tanto peligrosa por cuanto podria producir resultados impredecibles cuando su valor es indefinido; pero cambiar el valor de este contador en el programa resulta beneficioso para ciertos procesos ya que al cambiar el valor del puntero de la tupla corriente lo que se esta haciendo en realidad es convertir el modulo de lectura secuencial en un modulo de lectura directa, pero como este contador (TUPLANUMERO) trabaja como puntero a posiciones fisicas dentro del archivo, el programador debera asegurarse de la posicion fisica de cada tupla en el archivo tomando las debidas precauciones para aquellas tuplas eliminadas.

El modulo LECSECTU siempre leera la tupla apuntada por TUPLANUMERO; en donde las tuplas han sido numeradas desde 0 hasta N-1 tuplas, esto debe ser tomado en cuenta si se va realizar una lectura directa.

4.4 LECTURA DE ARCHIVOS INVERTIDOS Y DE CLASIFICACION

Procedimiento: LECDIRTU.

OBJETIVOS:

Este modulo ha sido implementado para permitir al usuario de la unidad Polirel accesar los archivos invertidos y/o de clasificacion creados por las UTILITARIAS del sistema; esto hara posible procesar un grupo de tuplas que cumplen con un criterio de seleccion dado o procesar todas las tuplas de una relacion en el orden determinado por el usuario.

FORMATO:

LECDIRTU(relacion,tabla-de-atributos,

codigo-de-seleccion)

donde:

Relacion: especifica el nombre de la relacion a procesar, este es una cadena de cuatro caracteres cuyo formato es:

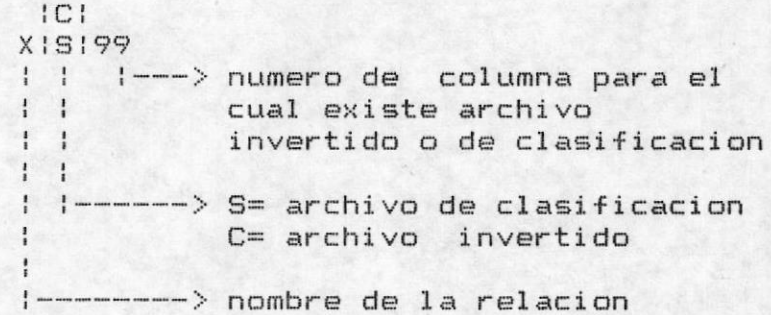


Tabla-de-atributos: igual que en LECSECTU define el formato de registro del usuario a utilizar en el proceso.

Codigo-de-seleccion: especifica una variable de tipo String, la cual identificara la posicion de la clave que sera utilizada como criterio de busqueda, el usuario debera colocar el valor de la clave de busqueda en la ocurrencia [1] de los arreglos respectivos (CAPOL, REPOL, ENPOL, STPOL) dependiendo de la estrategia de la clave; para identificar el arreglo en el cual ha sido depositado el valor de la clave el usuario almacenara en esta variable uno de los siguientes valores:

"CAPOL"

"ENPOL"

"STPOL"

"REPOL"

Para el caso de archivos de clasificacion no se necesita cargar el valor de la clave en ningun arreglo pero en codigo de seleccion se debe almacenar la palabra clave "$SORT".

DESCRIPCION:

Al invocar al procedimiento LECDIRTU se verificara siempre si la relacion motivo de la lectura este en memoria, la primera vez por lo tanto se invocara al

sistema multi-archivo para que proceda a cargar la respectiva relacion a memoria, se verificara ademas la existencia de los archivos invertidos y/o de sort necesarios para el proceso.

De igual manera se controlara que el criterio de seleccion sea al mismo en cada LECDIRTU, esto permitira al usuario procesar todas las tuplas que cumplen un criterio dado.

El procedimiento LECDIRTU por lo tanto manejara dos tipos de relaciones en ese momento, manteniendo en memoria los archivos de la relacion principal de donde se obtienen los datos para el usuario y los archivos invertidos y/o de clasificacion de donde se obtienen los punteros (numero de tupla) a las tuplas de la relacion principal, para lo cual se utiliza el metodo de lectura directa descrito en el topico anterior.

La lectura continua hasta detectar el fin de datos en el archivo invertido o de clasificacion emitiendose en dicho caso el OPCONDICION correspondiente.

Para obtener informacion tutorial y con ejemplos ver el capitulo respectivo en el manual operativo de la base de datos.

4.5 ELIMINACION DE TUPLAS

Procedimiento: ELIMINTU.

OBJETIVOS:

El modulo ELIMINTU ha sido implementado en la libreria del sistema para permitir al usuario de la Base de Datos eliminar aquellas tuplas que han dejado de ser utiles para asi mantener al dia el contenido de sus relaciones.

FORMATO:

```
ELIMINTU(relacion);
```

en donde:

Relacion: especifica el nombre de la relacion de la cual se eliminara la tupla, el nombre debe ser un caracter alfabetico y debe ser el mismo que se utilizo para un LECSECTU o LECDIRTU previo.

DESCRIPCION:

El procedimiento ELIMINTU eliminara la ultima tupla leida, es decir, para poder utilizar este procedimiento se debe haber leido una tupla anteriormente (utilizando el procedimiento LECSECTU o LECDIRTU).

Cada vez que este procedimiento es invocado se verificara que la relacion sea la misma a la que se le realizo un LECSECTU o LECDIRTU previo, en caso contrario

se cancelara este procedimiento emitiendo el OPCONDICION respectivo.

El procedimiento de lectura realizado anteriormente dejara en memoria actualizado el puntero a la tupla a ser eliminada, el proceso de eliminacion consiste en colocar un asterisco en la marca de eliminacion de la tupla (ultima posicion de la tupla corriente).

Al terminar se actualiza la variable que contiene el numero de tuplas existentes en la relacion asi como el arreglo utilizado por el sistema multi-bloque.

Importante:

Todo proceso de eliminacion de tuplas debe finalizar con una sentencia FINPROCESO para permitir a la unidad Polirel mantener la integridad de los datos contenidos en la base.

Para obtener informacion a detalle y con ejemplos ver Manual operativo de la Base de Datos.

4.6 ACTUALIZACION DE TUPLAS

Procedimiento: ACTUALTU.

OBJETIVO:

Este modulo ha sido implementado para permitir a los usuarios de Polirel mantener sus relaciones (archivos) actualizados y con informacion al dia.

FORMATO:

ACTUALTU(relacion,tabla-de-atributos);

donde:

Relacion: es el nombre de la relacion de la cual se actualizara la tupla previamente leida.

Tabla-de-atributos: especifica el diseno de registro definido por el usuario.

DESCRIPCION:

Se actualizara siempre la ultima tupla leida, es decir, que para poder utilizar este procedimiento se debe haber realizado un LECSECTU o un LECDIRTU previo a dicha relacion; este lectura previa es obligatoria ya que el diseno de registro definido por el programador en la lectura rige como formato para el proceso de actualizacion, por lo tanto siempre se validara que la relacion tenga esta lectura previa. En caso de encontrar

incoherencias en el proceso el procedimiento se cancelara emitiendose el valor del OPCONDICION correspondiente.

El proceso de actualizacion requiere que cada dato este en el arreglo adecuado y en la ocurrencia exacta, ademas que el programador tenga pleno conocimiento de la estrategia y posicion que ocupa cada columna dentro de la relacion dada.

Los nuevos valores deberan ser depositados en los arreglos respectivos (CAPOL, REPOL, STPOL, ENPOL) despues de ejecutar el procedimiento LECSECTU puesto que si se lo hace antes este ultimo los anulara.

El proceso de actualizacion se lo realiza columna por columna tomando las consideraciones que cada estrategia exige, especialmente para la actualizacion de cadenas de caracteres en aquellos casos donde la cadena nueva es mayor en longitud que la cadena anterior.

Al finalizar se actualizaran los nuevos punteros a los datos, su nueva longitud, el bloque donde se encuentran, etc.

Importante:

Todo proceso de actualizacion de tuplas debe finalizar con una sentencia FINPROCESO para permitir a la unidad polirel mantener la integridad de los datos contenidos

en la base.

Para obtener informacion a detalle y con ejemplos ver el manual Operativo de la Base de Datos.

4.7 ADICION DE TUPLAS

Procedimiento: ADICIOTU

OBJETIVOS:

Este modulo ha sido implementado para permitir que nuevas tuplas sean adicionadas a las ya existentes dentro de una relacion.

FORMATO:

```
ADICIOTU(relacion);
```

donde:

Relacion: especifica el nombre de la relacion a la que se adicionaran las tuplas siguientes y cuyos datos han sido dejados por el programador en los arreglos correspondientes.

DESCRIPCION:

Cuando el modulo ADICIOTU es invocado se activara un proceso de verificacion de la relacion en cuestion, si la relacion se encuentra ya en memoria se procedera a la insercion de la tupla correspondiente, en caso contrario el archivo debera ser abierto y leidos sus bloques respectivos para obtener los parametros de trabajo, luego se procede a retirar la estrategia y la posicion fisica de cada columna las cuales se almacenaran en

arreglos globales que serviran de guia para subsiguientes inserciones de tuplas ya que este proceso se lo realiza solo la primera vez.

El usuario debera colocar los valores de todos los atributos (columnas) de dicha relacion en los arreglos predefinidos en el interface de la unidad (ENPOL, CAPOL, STPOL, REPOL), dependiendo de las estrategias de los mismos, antes de invocar al procedimiento ADICIOTU; si el usuario no almacena ningun valor para un atributo dado, el procedimiento tomara el valor que tenga en ese momento la ocurrencia del arreglo correspondiente a ese atributo, todos los arreglos tienen valores iniciales (blanco para estrategia caracter y string, cero para las estrategias entero y real).

Es responsabilidad absoluta del usuario inicializar las ocurrencias de los arreglos con los valores de los atributos para asi evitar que el procedimiento tome los valores asumidos u otros datos dejados por el usuario en anteriores inserciones de tuplas.

La nueva tupla sera colocada a continuacion de la ultima existente en la relacion (archivo); por lo tanto habra un proceso de verificacion del espacio disponible en los archivos, en caso de no existir mas espacio disponible se cancelara el programa con los mensajes respectivos, si se ha determinado que se puede almacenar dicha tupla

se procedera a insertarla columna por columna en el orden que fueron definidas; los datos seran tomados de los arreglos correspondientes dependiendo de la estrategia de los mismos.

Cuando se ha terminado de insertar todas las columnas se actualizara el campo que contiene el numero de tuplas activas totales (NUMTUPLAS) y el arreglo de tuplas activas dentro de cada bloque para el proceso multibloque.

Importante:

Todo proceso de insercion de tuplas debe terminar con una sentencia FINPROCESO para permitir a la unidad Polirel mantener la integridad de los datos mantenidos en la base.

Para obtener informacion a detalle y con ejemplos ver manual operativo de la Base de Datos.

5. EDITOR

5.1 DEFINICION DE RELACIONES

- OBJETIVO:

  Descripcion de las columnas para la relacion.

- PARAMETROS:

  Entrada: Nombre de relacion y Descripcion de sus columnas.

  Salida : Archivos ADT y ADA actualizados.

- DESCRIPCION DEL PROCESO:

  Dado el nombre de la relacion se verifica si existen en el directorio los respectivos archivos (ADT,AT,ADA), tambien si ha sido definida previamente, en este caso existe una opcion para volverla a definir o no.

  El proceso de definicion de relaciones graba por cada columna su descripcion y al final del proceso informacion general de la relacion en el archivo ADT; ademas graba en el archivo ADA el nombre de las columnas una a continuacion de la otra.

  Los campos que se graban en el archivo ADT son los siguientes:

  - Descripcion de las columnas

. Estrategia

. Posicion en AT donde comienza el primer dato de la columna

. Numero del bloque en ADA donde se encuentra el nombre de la columna

. Posicion en ADA donde comienza el nombre de la columna

. Longitud del nombre de la columna

. Marca de eliminacion de la columna, en blanco

- Informacion general

. marca de eliminacion de relacion, en blanco

. Numero de tuplas

. Numero de columnas

. Longitud de la tupla

. Numero del bloque del archivo ADA que tiene la siguiente posicion disponible

. Siguiente posicion disponible en el archivo ADA

5.2 INGRESO DE TUPLAS

- OBJETIVO:

  Grabar tuplas para una relacion definida.

- PARAMETROS:

  Entrada: Datos de las tuplas a ingresar

  Salida : Archivo de la relacion con tuplas adicionadas

- DESCRIPCION DEL PROCESO.

  El proceso de ingreso de tuplas se realiza unicamente sobre una relacion existente. Se obtiene del archivo ADT, la longitud de la tupla y el numero de columnas de la relacion; luego para cada tupla que se desea ingresar se realiza un proceso tantas veces como columnas tenga la relacion.

  Este proceso para cada columna consiste en obtener la estrategia de la columna desde el archivo ADT, y dependiendo de ella el usuario ingresa el respectivo dato por pantalla, para luego ser validado y grabado.

  La grabacion de los datos se realiza de la siguiente manera:

  - Si el dato es entero, caracter o real, se lo graba en el archivo AT.

- Si el dato es una cadena de caracteres, su descripcion se graba en el archivo AT y el dato mismo en el archivo ADA.

Los siguientes campos en el archivo ADT son actualizados:

- numero de tuplas de la relacion.
- arreglo de tuplas logicas dentro de cada bloque
- Bloque y apuntador a la siguiente posicion disponible en el archivo ADA, para posteriores procesos.

El proceso de ingreso de tuplas graba los datos solo al final del proceso o cuando cambia el bloque ya sea del AT o del ADA.

Tambien se controla que exista espacio para ingresar los datos en los dos archivos anteriores, en caso de que no exista el suficiente espacio se emitira el respectivo mensaje.

5.3 ELIMINACION DE DATOS

- OBJETIVO:

  Permite la eliminacion de tuplas o columnas de una relacion creada

- PARAMETROS:

  Entrada: Archivos de relacion

  Salida : Archivo de la relacion modificado con marcas de eliminacion.

- DESCRIPCION DEL PROCESO:

  Este modulo se realiza unicamente sobre una relacion creada; se emplean los procedimientos LECSECTU y ELIMINTU de la Unidad Polirel (Ver Numerales 4.3 y 4.5).

  Para el proceso se obtiene del archivo ADT los siguientes campos a modificar:

  - numero de tuplas de la relacion
  - numero de columnas de la relacion

  Dependiendo de si se desea eliminar una tupla o columna se realiza lo siguiente:

  Si es una tupla:

  Dada la tupla logica a eliminar, se visualiza los

datos de dicha tupla, para lo cual se emplea el procedimiento LECSECTU. Si despues de ver los datos, se decide a eliminar esa tupla se emplea el procedimiento ELIMINTU, el que se encargara de grabar en el campo de marca de eliminacion de tupla, en el archivo AT, un asterisco y de decrementar el numero de tuplas logicas grabado en el archivo ADT.

Si es una columna:

Se visualiza todos los nombres de las columnas activas de la relacion y el usuario debera marcar con un "X" la(s) columna(s) a eliminar; si decide eliminar se realiza un calculo para obtener la posicion donde comienza la descripcion de la(s) columna(s) en el archivo ADT y en el campo de marca de eliminacion de la columna correspondiente se grabara un asterisco.

En caso de que se eliminare todas las columnas de la relacion, se grabara un asterisco en el campo que indica el estado de la relacion, lo que indicara que la relacion ha sido eliminada.

Observacion:

Al final del modulo de eliminacion se invocara al procedimiento FINPROCESO de la Unidad Polirel, el que se encargara de actualizar los bloques ADT, AT y ADT; y de cerrar los archivos.

Cabe recalcar que la eliminacion de tuplas y/o columnas de una relacion a traves de este modulo es solamente logica. Sin embargo, despues de ejecutar esta opcion, el usuario puede optimizar el espacio fisico de los archivos ejecutando las opciones COLECCION DE BASURA o DESCARGA Y CARGA DE RELACIONES, que se encuentran dentro del modulo UTILITARIAS (Ver Numerales 9.3, 9.5 y 9.6)

5.4 CORRECCION DE TUPLAS

- OBJETIVOS:

  Mantener los archivos ADT, AT y ADA actualizados mediante la correccion de cualquier dato de la relacion dada.

- PARAMETROS:

  Entrada: Archivos de la relacion, ingreso de datos correctos.

  Salida : Archivos de la relacion actualizados.

- DESCRIPCION DEL PROCESO:

  Se realiza exclusivamente sobre una relacion ya existente.

  Se accesa a la tupla requerida mediante el procedimiento LECSECTU (Ver Numeral 4.3.1), asignando previamente al campo TUPLANUMERO el numero de la tupla logica disminuido en 1.

  Dependiendo de la estrategia de la columna a corregir valida el ingreso del nuevo dato, si es una cadena de caracteres su longitud maxima sera 50.

  Cuando se termina de corregir todos los datos de una tupla se invoca al procedimiento ACTUALTU (Ver Numeral

4.6), actualizando asi los archivos correspondientes. Cuando se termina de corregir una relacion se invoca al procedimiento FINPROCESO para mantener la integridad de los datos contenidos en la Base.

5.5 CONSULTA DE TUPLAS

- OBJETIVO:

  Permite visualizar las tuplas de una relacion.

- PARAMETROS:

  Entrada: Archivos de la Relacion

  Salida : Despliegue/impresion de tuplas de la Relacion

- DESCRIPCION DEL PROCESO:

  Este modulo de la Base de datos permite al usuario realizar una consulta sobre cualquier relacion, utilizando el procedimiento LECSECTU de la Unidad Polirel, descrito en este mismo manual, Numeral 4.3.

  El modulo de consulta de la Base de datos presenta 2 opciones:

  1. Desplegar un relacion
  2. Listar una relacion

  El despliegue/listado de una relacion podra hacerse en dos formas:

  1. Tabla de tuplas
  2. Tupla por tupla

Tabla de Tuplas.-

Presenta los datos de una tupla en forma horizontal, tupla por tupla, maximo hasta 10 tuplas si es por consola; y hasta 50 tuplas por pagina si es por impresora.

Tambien le permite al usuario seleccionar las columnas a visualizar, marcando con una letra "X" las columnas que no desea ver; para el caso de una columna de estrategia string, se podra seleccionar desde 1 hasta 20 caracteres (se asume 10) del dato a visualizar.

Tupla por Tupla.-

Presenta los datos de una tupla en forma vertical, cada columna en una linea.

Para ambos casos existen ciertos comandos que facilitan el despliegue/listado de las tuplas, los mismos que se describen en detalle en el Manual Operativo del Sistema.

5.6 BUSQUEDA DE TUPLAS

- OBJETIVOS:

  Dado un criterio de busqueda; presenta por pantalla todas aquellas tuplas que satisfagan dicho criterio.

- PARAMETROS:

  Entrada: Archivos de la relacion, dato a buscar.

  SALIDA : Presentacion por pantalla de las tuplas que satisfagan la busqueda.

- DESCRIPCION DEL PROCESO:

  Se realiza exclusivamente sobre una relacion ya existente.

  Dependiendo de la estrategia de la columna por la cual se hace la busqueda, valida el ingreso del dato a encontrar.

  Se lee secuencialmente las tuplas de la relacion recuperando el dato de la columna dada. A continuacion se compara el dato a buscar con el de la tupla, si son iguales despliega toda la informacion de la tupla.

  Si el dato a encontrar es una cadena de caracteres, se busca la primera ocurrencia dentro de la cadena de caracteres de la tupla corriente.

Una vez presentada la primera tupla en pantalla, existe la opcion de seguir desplegando las demas tuplas que cumplan dicha condicion.

6. GRAMATICA

6.1 NOMBRES

6.1.1 NOMBRES DE RELACION:

Los nombres de relaciones estan constituidos por una letra (desde la 'A' hasta la 'Z')

6.1.2 NOMBRES DE SELECCION:

Los nombres de selecciones permanentes estan constituidos por dos letras: la primera que indica una estructura de seleccion y que debe ser siempre 'S'; y la segunda que puede ser una letra cualquiera (A-Z).

Los nombres de selecciones temporales estan constituidos por tres caracteres: el primero que indica una estructura de seleccion y que debe ser 'S'; y los otros dos son numeros que corresponden al numero de la entrada en la tabla de selecciones en que se encuentran definidos.

6.1.3 NOMBRES DE JOIN:

Los nombres de joins permanentes estan constituidos por dos letras: la primera que indica una estructura de Join y que debe ser 'J'; y la segunda una letra cualquiera (A-Z).

Los nombres de joins temporales estan constituidos

por tres caracteres: el primero que identifica una estructura de Join y que debe ser 'J'; y los otros dos son numeros que corresponden al numero de la entrada en la tabla de Joins en que se encuentran definidos.

6.1.4 NOMBRES DE PROYECCIONES:

Los nombres de proyecciones permanentes estan constituidos por dos letras: la primera que indica una estructura de proyeccion y que debe ser siempre 'P'; y la segunda que puede ser una letra cualquiera (A-Z).

Los nombres de proyecciones temporales estan constituidos por tres caracteres: el primero que identifica una estructura de proyeccion y que debe ser 'P'; y los otros dos son numeros que corresponden al numero de la entrada en la tabla de proyecciones en que se encuentran definidos.

6.2 OPERADORES

Los siguientes operadores son disponibles para las operaciones del QUERY:

:	Seleccion
*	Join
%	Proyeccion
LIST	Impresion en papel
DISP	Despliegue en pantalla

6.3 ESTRUCTURAS TEMPORALES Y PERMANENTES

6.3.1 TEMPORALES:

Toda operacion (Join, Seleccion, Proyeccion) que no contemple un enunciado de asignamiento sera considerada como una estructura temporal. Es decir, toda operacion que deba ser realizada como paso intermedio por el procesador para la obtencion de una operacion final; por el hecho de ser temporal, no podra ser utilizada por el usuario como paso inicial o intermedio en una linea de enunciado posterior a ser interpretado por el Query.

El Sistema asignara nombres a todas aquellas estructuras temporales generadas por operaciones del mismo tipo.

6.3.2 PERMANENTES:

Toda estructura y operacion para que sea considerada por el sistema como definitiva debera ser especificada explicitamente mediante un enunciado de asignamiento.

Toda estructura permanente podra ser utilizada en enunciados posteriores siempre y cuando sea permitido su uso segun las reglas sintacticas establecidas.

6.4 CLAUSULAS

6.4.1 SELECCION:

El operador seleccion trabaja exclusivamente sobre relaciones, permitiendo hasta dos criterios de seleccion relacionados mediante AND/OR. La estructura final puede ser permanente o temporal.

<relacion> : <criterio 1> AND/OR <criterio 2>

6.4.2 JOIN:

El operador Join trabaja sobre relaciones o selecciones, pero no puede haber un join entre dos selecciones. La estructura final puede ser permanente o temporal.

<relacion 1> * <relacion 2>

<seleccion> * <relacion>

<relacion> * <seleccion permanente>

6.4.3 PROYECCION:

El operador proyeccion trabaja sobre relaciones, joins y selecciones. La estructura final es permanente.

<relacion> % <columna 1>,<columna 2>..<columna N>

<seleccion> % <columna 1>,<columna 2>..<columna N>

<join> % <columna 1>,<columna 2>..<columna N>

6.4.4 LIST/DISP:

Estos operadores trabajan sobre estructuras permanentes solamente. LIST permite obtener un listado impreso de la estructura solicitada, DISP permite obtener un despliegue de la misma en la pantalla.

LIST <relacion>

LIST <seleccion>

LIST <join>

LIST <proyeccion>

DISP <relacion>

DISP <seleccion>

DISP <join>

DISP <proyeccion>

6.5 ASIGNAMIENTO

Un enunciado de asignamiento es utilizado para definir una estructura permanente y se especifica mediante el signo ':=' de la siguiente manera:

<seleccion> := <enunciado interpretado>
<join> := <enunciado interpretado>
<proyeccion> := <enunciado interpretado>

EJEMPLOS:

SA := A : CODIGO = 25

Seleccion permanente SA sobre la relacion A en que el campo CODIGO sea igual a 25.

JA := A * B

Join permanente JA sobre las relaciones A y B.

JA := A : CODIGO = 25 * B

Join permanente JA sobre la relacion B con la seleccion termporal de la relacion A, en donde el campo CODIGO sea igual a 25.

PA := A : CODIGO = 25 * B % CODIGO, NOMBRE

Proyeccion permanente PA de las columnas CODIGO y NOMBRE sobre el join temporal de la relacion B con la seleccion temporal de la relacion A, en que el campo CODIGO sea igual a 25.

PB := SA % CODIGO,CIUDAD

Proyeccion PB de las columnas CODIGO Y CIUDAD sobre la seleccion permanente SA.

6.6 FORMATO

El formato de un enunciado a ser interpretado por el preprocesador del QUERY es formato libre, es decir, puede contener todos los espacios que sean requeridos para mayor legibilidad del enunciado.

7. TABLAS Y ESTRUCTURAS

7.1 TABLA DE SELECCIONES

La Tabla de Selecciones esta definida asi:

```
TABSELECCIONES: ARRAY [1..MAXIMO] OF REGSELECCIONES;
```

Donde:

MAXIMO, es una constante igual a 25, que indica el numero maximo de selecciones que puede haber; y

REGSELECCIONES esta definido de la siguiente manera:

```
REGSELECCIONES =
 RECORD
  NOMBSEL:STRING[3];      Nombre de la seleccion
  NOMBREL:STRING[1];      Relacion de origen
  NUMCOL,                 Numero de columnas de la
                          relacion de origen.
  NUMTUP,                 Numero de tuplas de la relacion
                          de origen
  LONGTUP,                Longitud de la tupla de la
                          relacion de origen en el
                          archivo TF
  TUPSELEC:INTEGER;       Numero de tuplas seleccionadas
  FIRST,                  Apuntador al primer nodo de la
                          lista enlazada
  LAST: AP;               Apuntador al ultimo nodo de la
                          lista enlazada
 END;
```

En el caso de que la lista enlazada de tuplas seleccionadas no tenga nodos, los apuntadores FIRST Y LAST tendran los valores de NIL (nulo).

Los nodos de la lista enlazada estan definidos de la siguiente manera:

```
AP=P;                    Definicion del apuntador para
                         la lista enlazada

P = RECORD
    POSITION:INTEGER;    Posicion relativa de la tupla
                         seleccionada en el archivo TF.
    NEXT:AP;             Apuntador al siguiente nodo de
                         la lista enlazada
    END;
```

7.2 TABLA DE JOINS

La tabla de Joins esta definida de la siguiente manera:

```
TABLAJOINS: ARRAY [1..MAXIMO] OF REGJOIN;
```

donde:

MAXIMO, es una constante igual a 25, que indica el numero maximo de joins que puede haber; y

REGJOIN esta definido de la siguiente manera:

REGJOIN=

```
 RECORD
  NOMBREJOIN:STRING[3];   Nombre del join
  F1:STRING[1];           Nombre de la relacion 1
  NUMCOL1,                Numero de columnas de la
                          relacion 1
  LONT1:INTEGER;          Longitud de las tuplas de la
                          relacion 1 en el archivo TF
  F2:STRING[1];           Nombre de la relacion 2
  NUMCOL2,                Numero de columnas de la
                          relacion 2
  LONT2:INTEGER;          Longitud de las tuplas de la
                          relacion 2 en el archivo TF
  NTUP:INTEGER;           Numero de tuplas seleccionadas
                          para el Join
  MATCOL:STRING[6];       Nombre de la columna comun en
                          ambas relaciones
```

```
  FRENTE,                 Apuntador al primer nodo de la
                          lista enlazada
  ATRAS:POINTER;          Apuntador al ultimo nodo de la
                          lista enlazada
 END;
```

En el caso de que la lista enlazada de tuplas que forman el join no tenga nodos, los apuntadores FRENTE Y ATRAS tendran el valor de NIL (nulo).

Los nodos de la lista enlazada estan definidos de la siguiente manera:

```
POINTER=NODOJOIN;         Definicion del apuntador para
                          la lista enlazada
NODOJOIN=RECORD
        APREL1,           Posicion Relativa de la tupla
                          seleccionada para el
                          Join en el archivo TF de la
                          relacion 1
        APREL2:INTEGER;   Posicion relativa de la tupla
                          seleccionada para el
                          Join en el archivo TF de la
                          relacion 2
        NEXT:POINTER;     Apuntador al siguiente nodo de
                          la lista enlazada
        END;
```

7.3 TABLA DE PROYECCIONES

La tabla de proyecciones esta definida asi:

TABPROYECCIONES: ARRAY [1..MAXIMO] OF REGPROYECCIONES;

Donde:

MAXIMO, es una constante igual a 25 que indica el numero maximo de proyecciones que puede haber; y

REGPROYECCIONES esta definido de la siguiente manera:

```
REGPROYECCIONES=
 RECORD
  NOMBREPROYECCION,        Nombre de la proyeccion
  QUEPROYECTA:STRING[3];   Nombre de la estructura que se
                           quiere proyectar
  NUMCOLUM:INTEGER;        Numero de columnas proyectadas
                           que esta en la tabla
  COLUMNAS:ARRAY[1..MAXCOLUMNAS] OF
                           Tabla de columnas proyectadas
                           (Maxcolumnas=10)
  RECORD
   RELADECOL:STRING[1];    Relacion a la que pertenece la
                           columna proyectada
   NOMCOL:STRING[6];       Nombre de la columna a
                           proyectar
  END;
 END;
```

7.4 TABLA GENERAL

La tabla general de estructuras permanentes o tabla de simbolos esta definida de la siguiente manera:

```
TABVAR: ARRAY[1..MAXEST] OF TABVARPERM;
```

donde:

MAXEST, es una constante igual a 50, que indica el numero maximo de estructuras permanentes que puede haber; y

TABVARPERM esta definido de la siguiente manera:

```
TABVARPERM=
 RECORD
  VARIABLE:STRING[3];     Nombre de la estructura
                          permanente
  POSIC:INTEGER;          Numero de la entrada a la tabla
                          de la estructura respectiva;
 END;
```

8. QUERY

8.1 QUERY

El Query es el programa principal de la Base de Datos, por lo tanto, es donde se definen todas las constantes, tipos de variables y estructuras a ser utilizadas por los procesos de la Base de Datos y sus procedimientos generales.

La unica entrada de datos a la base es el enunciado a ser procesado.

El programa Query se divide en los siguientes modulos:

1. Analisis de Lexico
2. Analisis de Sintaxis
3. Interpretacion

El Enunciado a ser procesado por el Query debera entonces pasar por las etapas de analisis para su correspondiente interpretacion y procesamiento.

8.1.1 CONSULTAS CATALOGADAS

Si desde el Query se desea procesar un archivo de consultas, este lee dicho archivo. Luego toma las primeras 50 posiciones, chequea que la primera posicion no sea un asterisco ya que en ese caso seria una linea eliminada, la cual no se procesa. Chequea que la linea no este en blanco ya que esta es interpretada como fin de la consulta; si la primera posicion esta en blanco, transfiere las siguientes 49 posiciones al proceso normal del QUERY. Una vez que la expresion ha sido procesada, continua con la siguiente expresion, hace los chequeos correspondientes y asi sucesivamente hasta finalizar la consulta.

8.1.2 SECUENCIA DE EJECUCION

- Declaraciones del Programa:

  Constantes, tipos y variables.

- Declaracion de procesos de la Base:

  Seleccion, Join, Proyeccion, List-Disp.

  Son declarados como SEGMENT PROCEDURE y llamados por el compilador como archivo INCLUDE.

- Procedimiento de Inicializacion:

  . Sets de la Base:

    Letras, numeros, operadores para cada proceso

  . Tabla de variables permanentes

  . Variables booleanas

Mientras se desee procesar un enunciado:

- Lectura del Enunciado
- Analisis de Lexico
- Analisis de Sintaxis
- Interpretacion
- Proceso del Enunciado

8.1.3 AUDITORIA

- OBJETIVO:

  El proceso de Auditoria se realiza con el objeto de que una persona autorizada lleve un control sobre la sesion de trabajo realizada por el usuario dentro del Modulo Query.

- DESCRIPCION DEL PROCESO:

  Se genera un archivo(Ver Numeral 3.2.4) donde se graba solo aquellas expresiones que han sido correctamente procesadas por el Query.

  Utilizando la identificacion digitada por el usuario al inicio de la sesion, se procede a encriptar cada una de las consultas efectuadas a traves del Query.

  El archivo de Auditoria resultante puede ser procesado mediante un modulo de descripcion para propositos de chequeo posterior. Este archivo es creado cada vez que el usuario interactua con el Query.

8.2 INTERPRETADOR

8.2.1 REGISTROS SINTACTICOS

Los registros sintacticos estan definidos de la siguiente manera:

- REGISTRO SINTACTICTO DE LA SELECCION

```
RSINTSEL =
    RECORD
      NOMBSEL:STRING[3];    Nombre de la seleccion
      NOMBREL:STRING[1];    Relacion de origen
      ID1:STRING[6];        Nombre de columna 1
      COND1:STRING[1];      Condicion Relacional
      ID2:STRING;           Identificador que
                            se busca en la columna
      CONDLOG:STRING[3];    Condicion logica
      ID3:STRING[3];        Nombre de columna 2
      COND2:STRING[1];      Condicion relacional
      ID4:STRING;           Identificador que
                            se busca en la columna
    END;
```

- REGISTRO SINTACTICO DEL JOIN

```
RSINTJOIN=
    RECORD
      NOMJOIN,              Nombre del Join
      UNOJOIN,              Nombre de Relacion o
                            Seleccion
      DOSJOIN:STRING[3]     Nombre de Relacion o
                            Seleccion
      RELA1,                Nombre de la Relacion
                            fuente en UNOJOIN
      RELA2:STRING[1];      Nombre de la Relacion
                            fuente en DOSJOIN
    END;
```

```
- REGISTRO SINTACTICO DE LA PROYECCION

  RSINTPROY=

   RECORD

     NOMPROY,                Nombre de la proyeccion

     ESTRU:STRING[3];        Estructura a

                             proyectar

     NCOLUM:INTEGER;         Numero de columnas a

                             proyectar

     NCOLUMNAS:PACKED ARRAY[1..MAXCOLUMNAS] OF

                  STRING[6]; Nombre de las columnas a

                             proyectar

   END;
```

- REGISTRO SINTACTICO DE LIST-DISP

```
RSINTLDIS=
    RECORD
      OPER:STRING[1];          L=list o D=disp
      ESTRUC:STRING[2];        Estructura a listar
                               o desplegar
    END;
```

8.2.2 ANALISIS DEL LEXICO

- OBJETIVOS:

  Forma los elementos sintacticos atomicos para generar y actualizar los respectivos registros sintacticos.

- PARAMETROS:

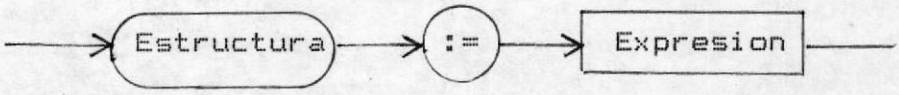
  Entrada: expresion o enunciado a procesar.

  Salida : registros sintacticos y tabla de secuencia de operadores actualizadas.

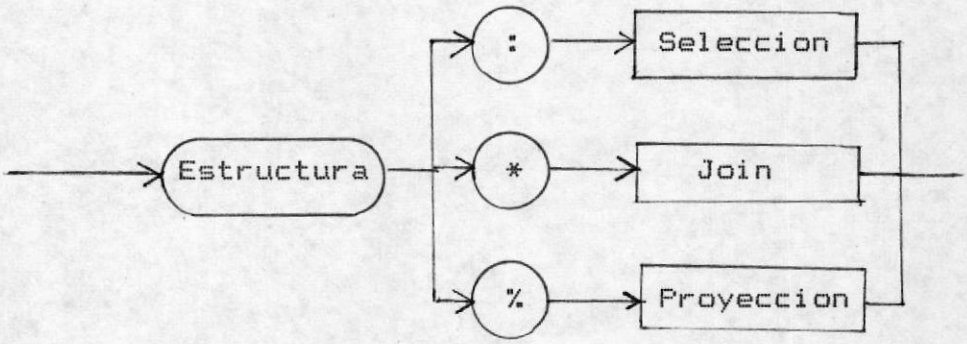
- DESCRIPCION DEL PROCESO:

  El proceso del lexico ha sido desarrollado a partir de los diagramas sintacticos (Ver Apendice F), transformando ciertos diagramas basicos en estructuras logicas manipuladas por el lenguaje Pascal como son el CASE, IF, etc.
  A continuacion se describe mediante un ejemplo, como se convierte el diagrama en una sentencia Pascal:

  Sentencia



Expresion



Donde expresion y sentencia se convierten en llamadas a procedimientos:

```
PROCEDURE SENTENCIA;
begin

end;
```

Dentro del procedimiento SENTENCIA, tenemos la siguiente estructura logica:

```
IF simbolo = ':=' THEN EXPRESION;
```

Dentro de expresion tenemos un grafico que

obliga a escoger uno de tres procesos dependiendo del operador; este grafico se resume en la siguiente estructura:

```
CASE  simbolo  OF
':' : SELECCION;
'*' : JOIN;
'%' : PROYECCION;
END;
```

Donde SELECCION, JOIN Y PROYECCION tambien son procedimientos que encierran llamadas sucesivas hasta llegar al procedimiento basico cuyas funciones las definiremos mas adelante.

El analisis del Lexico lo primero que realiza es la inicializacion de:

- Tabla de secuencia de operadores
- Registros Sintacticos (todos)
- Booleana de proceso.

Este modulo tiene un procedimiento basico que sirve para los siguientes propositos:

- Saltar blancos
- Reconocer palabras reservadas, tales como AND, LIST, etc.
- Reconocer palabras no reservadas, como son los

nombres de estructuras permanentes, nombres de columnas, constantes alfanumericas y numericas.

- Reconocer pares de caracteres especiales tales como ':='.
- Tambien identifica cada uno de los caracteres especiales que se utilizan ya sea como operadores (':','*','%','>') o separadores ('/',',').

Este procedimineto reconoce grupo de caracteres como un simbolo especifico (estructura, columna, operador de seleccion, etc).

El proceso de Lexico va rastreando la expresion de izquierda a derecha, dependiendo del simbolo corriente y de conformidad con los diagramas de sintaxis se actualiza la tabla de operadores y los registros sintacticos correspondientes. Ademas realiza un pre-chequeo sintactico de validaciones iniciales basicas.

8.2.3 ANALISIS DE SINTAXIS

El analisis de sintaxis se realiza con la ayuda de los registros sintacticos que fueron grabados por el analisis de lexico para el proceso corriente.

El proceso empieza por barrer la tabla de operadores, desde el primer operador hasta encontrar el operador que indica el final de la tabla, representado por el signo "&".

Dependiendo del operador de proceso se realiza el chequeo o validacion de datos del registro sintactico para ese proceso.

```
CASE operador[I] OF
  opselec      : CHEQUEOSELECCION(CORRECTO);
  opjoin       : CHEQUEOJOIN(CORRECTO);
  oppproy      : CHEQUEOPROYECCION(CORRECTO);
  oplit,oplist: CHEQUEOLDIS(CORRECTO);
END;
```

Si el chequeo de datos fue correcto se procede a la interpretacion del proceso, sea este temporal o permanente; caso de error se cancela la sintaxis emitiendo el respectivo mensaje de error.

Chequeo o validacion de datos significa:

- Validar los nombres en su terminologia usada

- Verificar si existen las estructuras de origen del proceso como relaciones, columnas, variables permanentes, etc.
- Verificar que el nombre no exista en la Tabla General.

CHEQUEO SELECCION.-

El chequeo de registro sintactico de la seleccion tiene la siguiente secuencia:

- Chequeo de nombres:

  Se chequea si el nombre dado como variable es valido.

- Existencia de Relacion:

  Se chequea si los archivos de la relacion de origen de la seleccion han sido creados.

- Existencia de columna:

  Se chequea si la columna identificador de la seleccion existe en la relacion de origen.

  Se obtiene la estrategia de la columna.

  Si la seleccion es compuesta, se repite el procedimiento para la segunda parte.

- Chequeo de Estrategia:

  Se chequea si los identificadores de la seleccion son compatibles con la estrategia de la columna, es decir, no es compatible la

expresion:

SUELDO > ROJO

Si la seleccion es compuesta se repite el procedimiento para la segunda parte.

Cualquier procedimiento que encuentre un error enviara el mensaje de error respectivo.

CHEQUEO JOIN.-

El chequeo del registro sintactico del join tiene la siguiente secuencia:

- Chequeo de nombres:

  Se valida si el nombre dado como variable permanente es valido.

  Si una de las estructuras fuentes del join es una seleccion permanente se chequeara si existe en la Tabla General de la Base.

- Determinacion de las estructuras fuentes del join:

  - Relacion, Seleccion:

    Si una de las estructuras fuentes es una seleccion busca la relacion de origen de la seleccion en la Tabla de selecciones.

- Existencia de Relacion:

  Se chequea si los archivos de las relaciones de origen del Join han sido creadas.

- Chequeo de la existencia de columna comun en las relaciones de origen.

Cualquier procedimiento que encuentre un error enviara el mensaje respectivo.

CHEQUEO PROYECCION.-

El chequeo del registro sintactico de proyeccion tiene la siguiente secuencia:

- Chequeo de nombres:

  Se chequea si el nombre dado como variable permanente es valido.

  Si la proyeccion es de una variable permanente se chequeara si existe en la tabla general de la Base.

- Determinacion de la estructura de origen de la proyeccion: Relacion, Seleccion, Join.

- CASE estructura de origen OF

  Relacion : Proceso de Validacion de columnas

  Seleccion: Busca la relacion de origen de la seleccion en la tabla de selecciones.

  Proceso de validacion de columnas.

  Join : Busca las dos relaciones de origen del Join.

  Proceso de validacion de columnas.

  Las columnas que no existan en la

primera relacion seran buscadas en la segunda relacion.

- Se verificara que las columnas a proyectarse existan en la(s) relacion(es) fuente(s).

Cualquier procedimiento que encuentre un error enviara el mensaje de error respectivo.

CHEQUEO LIST-DISP.-

El chequeo del registro sintactico del List-Disp tiene la siguiente secuencia:

- CASE estructura de origen OF

  Relacion : Existen archivos de la relacion de origen.

  Variable permanente: Busca en la tabla general si existe.

Cualquier procedimiento que encuentre un error enviara el mensaje de error respectivo.

8.2.4 INTERPRETACION

El proceso de interpretacion se realiza con la ayuda de los registros sintacticos que fueron validados en el Analisis de Sintaxis. Este proceso empieza por barrer la tabla de operadores, dependiendo del operador se ejecutan los diferentes procesos:

```
CASE OPERADORES[I] OF
 ':'     : PROCESOSELE (REGSEL,CORRECTO);
 '*'     : PROCESOJOIN (REGJOIN,CORRECTO);
 'L','D': PROCESOLDIS (REGLDIS);
END
```

8.3 PROCESO DE SELECCION

- PARAMETROS:

  Entrada: Registro sintactico de la seleccion

  Salida: Booleana del proceso de seleccion

  Actualizacion de la tabla de selecciones

- TABLA DE SELECCIONES:

  La descripcion del registro sintactico se encuentra en el Numeral 7.1. Cada entrada a la tabla contiene los datos y estructuras basicas de un proceso de seleccion, sea este temporal o permanente.

  Esta informacion estara disponible pra cualquier otro proceso que pueda requerirla.

  Cada vez que el proceso de seleccion es realizado exitosamente se incrementa en uno el contador de selecciones, para ingresar la informacion de la nueva seleccion en la Tabla General.

- DESCRIPCION DEL PROCESO DE SELECCION

  El objetivo de este proceso es crear una estructura que contenga todas aquellas tuplas dentro de una Relacion, que satisfacen las condiciones especificadas.

Se realiza a partir de Relaciones solamente. La seleccion puede realizarse a traves de dos procesos diferentes: uno que consiste en recorrer todo el archivo de tuplas (AT) para realizar el seleccionamiento de las mismas y, otro que utiliza los archivos de Indices invertidos para accesar directamente a las tuplas que deben seleccionarse.

- PROCESO SIN INDICES INVERTIDOS

  Se empieza por hacer una validacion de datos para la seleccion. El procedimiento principal de la seleccion busca las posiciones de las tuplas en el archivo de tuplas (AT), el numero de tuplas de la relacion, longitud de las tuplas y offset de las columnas en el archivo AT para la ejecucion del lazo de seleccion de tuplas.

  El lazo principal realiza el seleccionamiento de las tuplas que satisfacen los criterios dados hasta el numero de tuplas de la relacion.

  Los datos de las columnas son leidos del archivo de tuplas (AT) y asignados como identificadores. La comparacion de la seleccion se realiza con una o dos instrucciones CASE, si la seleccion es compuesta.

```
EXITOSO:=FALSE;
CASE CONDICION OF
    '>': IF ID1 > ID2 THEN EXITOSO:=TRUE;
    '<': IF ID1 < ID2 THEN EXITOSO:=TRUE;
    '=': IF ID1 = ID2 THEN EXITOSO:=TRUE;
```

Todos los identificadores son tomados como operandos tipo string. La seleccion es exitosa cuando la variable booleana exitoso es true.

Si la seleccion es compuesta la booleana debera ser true para ambos criterios.

Si la seleccion es exitosa se aumenta un nodo a la lista enlazada de selecciones y se incrementa el contador de tuplas exitosas.

- PROCESO CON INDICES INVERTIDOS

Despues de hacer la validacion de datos para la seleccion, para realizar la seleccion utilizando Indices invertidos deben cumplirse las siguientes condiciones:

- Que exista el numero de la Columna en el archivo ADT senalado entre las posiciones 400 a 407 y el archivo de Indices para ese atributo.
- Sea la expresion simple o compuesta, los criterios de seleccion deberan ser siempre por igual (=).

- Si la expresion es compuesta y el operador logico es "OR", deberan existir Indices para los dos atributos sobre los que se va a seleccionar.
- Si la expresion es compuesta y el operador logico es "AND", debera existir Indice para uno de los atributos sobre los que se forma el criterio de seleccion.

Bajo estas condiciones, se buscara dentro del bloque directorio del archivo de Indices el valor del atributo que satisface la seleccion y una vez encontrado, se procedera a anadir a la lista enlazada todos los numeros relativos de tuplas que se encuentren en el bucket o los buckets asociados a ese valor.

La comparacion para la seleccion se realiza con uno o dos IF...THEN, dependiendo de que la seleccion sea simple o compuesta.

```
EXITOSO=FALSE;
IF VALORATRIBUTO=ID2 THEN EXITOSO=TRUE;
```

Si la seleccion es compuesta entonces, si el operador logico es "OR", para el segundo criterio se

debera verificar en el momento en que se vaya a anadir un nodo a la estructura enlazada de la seleccion que el numero relativo de la tupla no se encuentre en ella, de lo contrario no se lo adiciona.

Si el operador logico es "AND", entonces una vez que se toma el numero relativo de la tupla del Archivo de Indices, se debera verificar accesando al archivo de tuplas, y de datos alfanumericos si es necesario, que la segunda condicion tambien se cumpla. Si es asi, se anadira un nodo mas a la cola, aumentando el contador de tuplas seleccionadas. Este proceso se repetira para cada una de las tuplas que son pre-seleccionadas en base al indice invertido de uno de los atributos.

Una vez que el proceso de seleccion de tuplas ha terminado, por cualquiera de los dos procesos, se registra en el descriptor de la estructura enlazada de la seleccion toda la informacion general sobre la misma.

8.4 PROCESO DE JOIN

- PARAMETROS:

  Entrada: Registro Sintactico del Join

  Salida: Booleana del proceso de Join

  Actualizacion a la tabla de Joins

- TABLA DE JOINS:

  La descripcion del registro se encuentra en el punto 7.2. Cada entrada a la tabla contiene los datos y estructuras basicas que son resultado de un proceso de Join, sea este temporal o permanente.

  Esta informacion estara disponible para cualquier otro proceso que pueda requerirla.

  Cada vez que el proceso de Join es realizado exitosamente se incrementa en uno el contador de Joins, para ingresar la informacion del nuevo Join.

- DESCRIPCION DEL PROCESO DE JOIN:

  El proceso de Join se realiza a partir de:

  1) Dos relaciones

     A * B

  2) Una seleccion permanente o temporal y una relacion

     SA * B, o A:CIUDAD = GQUIL * B

Cualquiera que sea su origen los datos y estructuras resultantes son exactamente los mismos.

El primer modulo del procedimiento valida la informacion ingresada: nombre de join y estructuras para formarlo.

En el siguiente paso se establece la columna en comun de ambas estructuras, la misma que va a servir para establecer las tuplas de las estructuras fuentes que van a formar parte del Join resultante.

Se procede a barrer ambas estructuras secuencialmente tomando como pivote aquella de mayor numero de tuplas en el caso de un Join entre 2 relaciones; si se trata de una seleccion con una relacion, el pivote sera siempre la relacion.

Cada vez que se encuentre una coincidencia de valor de los datos de las 2 estruturas fuentes en la columna comun, se procede a adicionar un nodo a la lista enlazada con los apuntadores a las relaciones formantes.

Cabe aclarar que en el caso de la seleccion, la lista enlazada que la forma servira solo para, con el apuntador del nodo corriente, accesar a los archivos de la relacion de origen, por lo tanto es este apuntador a la tupla el que se almacenara en el nodo

que forma el Join.

Cuando se acaban de barrer completamente las dos estructuras, se termina el proceso dejando ya formada la lista enlazada como resultado del Join.

8.5 PROCESO DE PROYECCION

- PARAMETROS:

  Entrada: Registro Sintactico de Proyeccion

  Salida : Booleana del Proceso de Proyeccion

           Actualizacion a la Tabla de Proyecciones

- TABLA DE PROYECCIONES:

  La descripcion del registro se encuentra en el punto 7.3.

  Cada entrada a la tabla contiene los datos y estructuras basicas que son resultados de un proceso de proyeccion, sea esta temporal o permanente.

  Esta informacion estara disponible para un proceso LIST/DISP posterior solamente.

  Cada vez que el proceso de proyeccion es realizado exitosamente se incrementa en 1 el contador de proyecciones, para ingresar la informacion de la nueva proyeccion.

- DESCRIPCION DEL PROCESO DE PROYECCION:

  El proceso de proyeccion se realiza a partir de:

  1) Una relacion

  2) Una seleccion temporal o permanente

3) Un Join temporal o permanente

Cualquiera que sea su origen los datos y estructuras resultantes son exactamente los mismos.

El proceso de Proyeccion comienza pasando los nombres de las columnas que se quiere proyecctar a una tabla, la que sera barrida buscando la posicion de la descripcion de cada columna en el archivo descriptor de tuplas (ADT) de la relacion fuente.

A continuacion se establece si la proyeccion a realizar es sobre una relacion, una seleccion o un join. Si la proyeccion es sobre una seleccion o un join se encuentran en la tabla de selecciones o joins, respectivamente, la relacion fuente, en el caso de la seleccion o las relaciones fuentes, en el caso del join.

En el caso de la proyeccion de una relacion, como es obvio, no es necesario buscar la relacion fuente.

Una vez encontrada la relacion fuente, si la proyeccion es sobre una relacion o una seleccion se lee los archivos de la relacion fuente (ADT, AT y ADA); se obtiene el numero de columnas que tiene la relacion fuente; se barre el archivo descriptor de tuplas buscando la posicion donde comienza la descripcion de cada una de las columnas que se quiere proyectar.

Si la proyeccion es sobre un join se realiza el proceso descrito, pero dos veces, uno por cada relacion fuente sobre los cuales fue realizado el join.

8.6 PROCESO DE LIST / DISP

- PARAMETROS:

  Entrada: Registro Sintactico de List-Disp

  Salida : Despliegue/impresion de relacion o
           estructura permanente

- DESCRIPCION DEL PROCESO:

Este modulo de la base de datos Polirel, permite al usuario listar o desplegar cualquier relacion, seleccion, join y/o proyeccion que esten en la tabla de estructuras permanentes.

Este modulo de List-Disp utiliza los procedimientos de la unidad Polirel: LECSECTU y READFISIC.

Para el caso de join o proyeccion de tuplas se presenta en forma vertical (una tupla a la vez).

Para el caso de relacion y seleccion de una relacion se podra escoger la forma de presentar las tuplas de la relacion.

1.- tabla de tuplas

2.- por una tupla

1. Tabla de tuplas:

   Presenta los datos de una tupla en forma horizontal, tupla por tupla, maximo hasta diez tuplas si es por consola, y hasta 50 tuplas por pagina si es por impresora. Tambien le permite al usuario seleccionar las columnas a visualizar, marcando con una letra X las columnas que no desea ver; para el caso de una columna de estrategia tipo string, se podra seleccionar desde 1 hasta 20 caracteres (asume 10 caracteres) del dato a visualizar; para salir de esta seleccion de columnas se debera digitar la letra Q.(Para mas detalle ver Numeral 5.5, CASO 1.)

2. Por una tupla:

   Presenta los datos de una tupla en forma vertical, cada columna en una linea. (Para mas detalle ver Numeral 5.5, CASO 2). Para la visualizacion de tuplas existen las siguientes opciones:

P (proximo	permite ver la siguiente tupla.
A (atras	permite ver la tupla anterior.
I (inicio	permite ver la tupla inicial.
R (relativo	dado un numero relativo de tupla logica, permite ver dicha tupla.
S (salir	retorna al menu del query.

SOBRE UNA RELACION:

Se busca en la tabla de selecciones la posicion de la seleccion sobre la cual se desea hacer List-Disp, para obtener la siguiente informacion.

a) nombre de la relacion.

b) lista enlazada cuyos nodos contienen el numero de tupla fisica de la relacion; se procesan los nodos empleando los procedimientos LECSECTU y READFISIC.

SOBRE UN JOIN:

Se busca en la tabla de joins la posicion del join sobre el que se desea hacer List-Disp, para obtener la siguiente informacion:

a) nombre de las relaciones sobre las que fue realizado el join.

b) lista enlazada cuyos nodos contienen:

- posicion relativa de la tupla de la primera relacion
- posicion relativa de la tupla de la segunda relacion
- nombre de la columna comun de ambas relaciones

Formando estas dos tuplas una sola tupla-join; se procesa esta tupla-join, empleando los procedimientos LECSECTU y READFISIC de la Unidad Polirel. La columna en comun solo se visualiza una vez.

SOBRE UNA PROYECCION:

Se busca en la tabla de proyecciones la posicion de la proyeccion sobre la que se desea hacer List-Disp, para obtener la siguiente informacion:

a) nombre de la relacion de origen de la proyeccion
b) nombre de la seleccion
c) nombre del join
d) numero de columnas a proyectarse
e) nombre de las columnas a proyectarse

Se procede en forma similar al proceso de una relacion, de una seleccion o de un join, desplegando/imprimiendo solo las columnas a proyectarse.

9. UTILITARIAS

9.1 FORMATEAR ARCHIVOS

- OBJETIVO:

  Creacion fisica de los Archivos (ADT, AT, ADA) para cada relacion.

- PARAMETROS:

  Entrada: Nombre de la Relacion y Numero de bloques para los archivos AT y ADA.

  Salida : Archivos de Relacion formateados.

- DESCRIPCION DEL PROCESO:

  Se verifica si existen en el directorio los archivos para la relacion, si se encuentran, aparecera una opcion para eliminarlos o no.

  Una vez que se acepta el numero de bloques tanto para el archivo AT como para el ADA, se formatean bloque a bloque cada uno de los archivos.

  Durante el proceso se graban en el AT los siguientes campos:

  - Se marca con una "F" el status de la relacion
  - Numero de bloques para el archivo AT y ADA

9.2 INDICES INVERTIDOS

Son archivos que como su nombre lo indica constituyen un indice de la relacion dada y se generan en virtud del contenido de un atributo (columna) particular.

- OBJETIVO:

  Este tipo de indices es aplicado para realizar una seleccion directa de aquellas tuplas que satisfagan un valor determinado para un atributo especifico.

- PARAMETROS:

  Entrada: Nombre de la Relacion, Numero de columna sobre la que se genera el indice.

  Salida : Archivo de Indices Invertidos

- DESCRIPCION DEL PROCESO:

  Despues de validar que existan los archivos de la relacion dada, se calcula el numero de bloques para el archivo de indices en base al numero de tuplas que tiene la relacion y al numero de valores diferentes dentro de esa columna (atributo).

  El archivo es formateado grabando tantos bloques en

blanco como se haya calculado previamente. Luego se registra en el archivo ADT el numero de la columna sobre la que se va a generar el indice.

Para cada relacion pueden generarse un maximo de 4 indices invertidos. Para cada indice perteneciente a un atributo existira un archivo. Cada vez que se genera un indice, este reemplazara a otro creado anteriormente, solo si se refiere al mismo atributo.

Una vez inicializado el bloque directorio con el numero total de Buckets, se procede a barrer el archivo AT de la relacion y a grabar en el directorio el valor del atributo, si este no existe ya, asignandole a cada entrada un bucket, en donde se grabaran los numeros relativos de todas aquellas tuplas que tengan dicho valor en el atributo seleccionado y actualizando el numero de buckets disponibles y las posiciones tanto en el bucket respectivo como en el bloque directorio.

Si se da el caso de que para un valor determinado existen mas de 20 tuplas (Ver descripcion de registro, Numeral 3.2.1), entonces se debera encadenar al bucket lleno otro que este disponible; grabando en las posiciones 1-2 del bucket lleno el numero del bucket que se encadena y en las posiciones 3-4 '00', de modo que indique que el bucket no tiene mas posiciones

disponibles.

La numeracion que tienen los buckets para su identificacion esta dada por su posicion relativa, es decir que el bucket 4 estara ubicado en el bloque 1, el bucket 16 sera el ultimo bucket del bloque 2, etc.

La longitud del valor del atributo que se registra en el directorio dependera de la estrategia del atributo (Ver Numeral 3.2.1), sin embargo para el caso de cadenas de caracteres, se tomaran unicamente los 10 primeros caracteres del valor del atributo (sin considerar espacios en blanco) para la generacion del indice; por lo tanto el numero de posibles entradas (valores diferentes del atributo) en el bloque directorio tambien variara de acuerdo a la estrategia:

ESTRATEGIA	# MAX.ENT.
'C'	99 (*)
'E'	73
'R'	56
'S'	42

(*) Cuado se trata de caracteres el numero maximo de entradas, no se encuentra restringido por la longitud del campo, sino por el numero maximo de tuplas que una Relacion puede tener.

Una vez generado el archivo de indices, este queda a disposicion del Proceso de Seleccion dentro del Query

para realizar en forma directa el seleccionamiento de las tuplas; del Generador de Reportes cuando se utilizan criterios de selecion; y de cualquier usuario de la Base de Datos, que interactue con ella a traves de la Unidad Polirel.

9.3 COLECCION DE BASURA

- OBJETIVOS:

  Compactar el ADA mediante la extraccion de las cadenas de caracteres eliminadas para optimizar el espacio disponible en este archivo.

- PARAMETROS:

  Entrada: archivos de la relacion.

  Salida: archivos de la relacion optimizados.

- DESCRIPCION DEL PROCESO:

  Se realiza exclusivamente sobre una relacion ya existente que tenga por lo menos una columna tipo string.

  Se lee el archivo ADT; se inicializa un apuntador al archivo ADA con la primera posicion disponible para cadenas de caracteres, esto es, la siguiente posicion despues del nombre de la ultima columna activa.

  Luego se va analizando cada tupla activa del archivo AT y se va chequeando la estrategia de cada columna en ADT, cuando se encuentra una cadena de caracteres chequea si esta activa la columna, en este caso toma de AT el numero de bloque, la posicion y la longitud

la posicion de una cadena de caracteres mas su longitud sea mayor que 512, en ese momento se graba este bloque y se lee el siguiente para proceder a llenarlo desde la primera posicion y asi sucesivamente hasta terminar con la lista enlazada.

Finalmente se actualiza en ADT el numero de bloque del archivo ADA que tiene la siguiente posicion disponible. Se regraba el archivo ADT y con esto termina el proceso.

de la cadena de caracteres , si no esta en memoria el bloque correspondiente de ADA, lo lee, procede a llenar un nodo. Los nodos de la lista enlazada estan definidos de la siguiente manera:

```
TYPE
NODO = RECORD
    LONG : INTEGER              Longitud de la cadena
                                de caracteres
    ADASTRING : STRING[50]      String de hasta 50
                                caracteres
    SIGNODO : ^NODO             Apuntador al siguiente
                                nodo de la lista
                                enlazada
END
```

Ademas va actualizando el numero de bloque, posicion y longitud de la cadena de caracteres en AT. Cada vez que se alcanza el numero de tuplas que entran en un bloque de AT, se regraba este y se lee el siguiente bloque de AT.

Una vez que se ha realizado este proceso para todas las tuplas de la relacion, se actualiza el archivo ADA y siguiendo los nodos de la lista enlazada,va llenando el primer bloque de ADA a partir de la posicion con que estaba inicializado el apuntador a ADA, hasta que

9.4 CLASIFICACION DE RELACIONES

- OBJETIVOS:

  Obtener un archivo de clasificacion en orden ascendente de una relacion dada, en virtud de una llave primaria (numero de columna) y si se requiere de una llave secundaria.

- PARAMETROS:

  Entrada: Nombre de la Relacion, Numero de la columna para Llave Primaria, Numero de columna para Llave Secundaria.

  Salida: Archivo clasificado.

- DESCRIPCION DEL PROCESO:

  Despues de validar que existan los archivos de la relacion dada, se formatea el archivo de clasificacion con un bloque en blanco (si existe el archivo, este queda eliminado).

  Por cada columna de la relacion dada se puede crear un archivo clasificado.

  Una vez inicializado el bloque se procede a barrer el archivo AT de la relacion, si la estrategia de la llave primaria es una cadena de caracteres, solo se

consideraran los primeros 20 caracteres; si existe llave secundaria se concatenaran los datos de las 2 columnas.

A continuacion se crea un nodo de la siguiente forma:

```
TYPE
  APUNTADOR = ^NODO;
  NODO = RECORD
    DATON  : STRING;     dato por el que se clasifica
    NUMTUP : INTEGER;    numero relativo de tupla
    APUNTI : APUNTADOR; apuntador al hijo izquierdo
    APUNTD : APUNTADOR; apuntador al hijo derecho
  END;
```

y se agrega al arbol en la forma recursiva del INORDER.

Cuando se termina de barrer el AT, se procede a atravesar el arbol y por cada nodo se grabara el numero relativo de la tupla ocupando 3 posiciones a partir de la quinta posicion en el archivo de clasificacion (Ver descripcion del archivo Numeral 3.2.2).

Si existen mas de 20 tuplas, entonces se debera encadenar al bucket lleno el siguiente disponible , guardando en las posiciones 1-2 del bucket lleno el numero del bucket encadenado y en las posiciones 3-4 '00', de modo que indique que el bucket no tiene mas

posiciones disponibles.

Si en el ultimo bucket utilizado hay posiciones libres; en las posiciones 3-4 de ese bucket, se guardara la siguiente posicion disponible dentro del bucket. Caso contrario, en las posiciones 1-2 se guardara el numero del siguiente bucket a encadenarse. Por ultimo se marca una 'X' en el archivo ADT en la posicion correspondiente a la columna considerada como llave primaria (Ver Numeral 3.1.1) para indicar que existe el archivo de clasificacion.

Una vez generado el archivo, este queda a disposicion del Generador de Reportes y de cualquier usuario de la Base de Datos, que utilize la Unidad Polirel (Ver Numeral 4.4).

9.5 DESCARGA DE RELACIONES

- OBJETIVO:

  Reorganizacion fisica de relaciones.

- PARAMETROS:

  Entrada : Nombre de relacion.

  Salida : Archivo de Reorganizacion para relacion.

- DESCRIPCION DEL PROCESO:

  Este proceso contempla la creacion de un archivo para una futura reorganizacion con la suficiente informacion para volver a generar nuevos archivos ADT, AT, ADA optimizados.

  Empieza verificando la existencia de los archivos para la relacion y que tenga por lo menos una columna habil.

  Luego verifica si ya existe un archivo de reorganizacion; si existe aparecera una opcion para eliminarlo o no.

  El proceso solo contempla aquellas columnas y tuplas que no estan eliminadas, generando de esta misma forma el archivo de reorganizacion.

  Se lee del archivo ADT toda la informacion que

corresponda a las columnas habiles como son la estrategia, numero de bloque en ADA, posicion en ADA, longitud de la columna, etc; para grabarla en el bloque directorio del archivo de reorganizacion y los nombres de las columnas separados por un '&' son grabados en el segundo bloque (Ver Numeral 3.2.3) y si la relacion tiene tuplas se graba secuencialmente cada uno de los datos tambien separados por un '&'.

9.6 CARGA DE RELACIONES

- OBJETIVO:

  Reorganizacion fisica de Relaciones.

- PARAMETROS:

  Entrada : Nombre de Relacion.

  Salida : Archivos para relacion reorganizados.

- DESCRIPCION DEL PROCESO:

  Con el nombre de la relacion se verifica si existe un archivo de reorganizacion, luego se lee el primer bloque del archivo de reorganizacion y se actualiza el archivo ADT de la relacion correspondiente.

  A continuacion se lee cada uno de los datos en los bloques siguientes y dependiendo de su estrategia se actualizan los archivos AT y ADA. Este proceso es similar al que Define Relaciones e Ingresa Tuplas, con la unica diferencia que es transparente para el usuario el ingreso de los datos, ya que estos han sido grabados en el archivo de Reorganizacion.

  Importante:

  Cuando los nuevos archivos AT y ADA han sido formateados con un menor numero de bloques al que

requiere la informacion contenida en el archivo de reorganizacion, el proceso grabara solo lo que alcance en los nuevos archivos AT y ADA, emitiendo los respectivos mensajes al usuario.

9.7 CONSULTAS CATALOGADAS

- OBJETIVO:

  Permitir almacenar y modificar consultas que puedan ser procesadas por el Query cada vez que se desee.

- DESCRIPCION:

  Consultas catalogadas tiene un procedimiento que se hace necesario mencionar y es el procedimiento PIDELINEA, el cual se utiliza en la mayoria de los procesos y este se encarga de pedir el numero de linea que el usuario desea eliminar, activar, modificar, consultar o insertar. Valida que esta linea no este en blanco, esto es, que exista la linea solicitada y la despliega.

  La localizacion de la linea deseada se realiza multiplicando el numero de linea menos uno por cincuenta que es la longitud maxima que se asigna a cada linea o expresion y mas uno, con esto se localiza la primera posicion de la linea deseada.

  Consultas catalogadas esta formada por cuatro procesos principales que los vamos a analizar por separado y que son:

  -Eliminacion

-Creacion

-Modificacion

-Consulta

ELIMINACION.-

Parametro de entrada: archivo de consulta catalogada

Proceso:

Obtiene el nombre del archivo a ser eliminado y lo borra fisicamente del directorio.

CREACION.-

Parametro de entrada: nombre de la consulta catalogada
Parametro de salida: archivo de consulta catalogada

Proceso:

Blanquea un arreglo de 512 caracteres, acepta una expresion, chequea que esta primera linea que acepta no este en blanco ya que no permite crear una consulta en blanco, deja la primera posicion del arreglo en blanco y mueve la expresion digitada a las 49 posiciones siguientes, y asi sucesivamente, va pidiendo la siguiente expresion, deja un blanco en el arreglo y pasa la expresion a las siguientes 49 posiciones del arreglo, el proceso se repite hasta que

la linea digitada este en blanco o ya se hayan digitado 10 lineas; luego el archivo es grabado e incorporado al directorio.

MODIFICACION.-

Parametro de entrada: archivo de consulta catalogada

Parametro de salida: archivo de consulta catalogada actualizado

La modificacion esta formada por 4 procesos, cada uno de los cuales al iniciarse llama al procedimiento PIDELINEA y cuando termina el procedimiento de modificacion se regraba el archivo de consulta.

1.- Modificacion de una linea.-

Permite escribir la nueva linea, la cual pasa a reemplazar a la anterior expresion, esto es, se mueve la nueva linea a las mismas 49 posiciones asignadas para la linea que se esta modificando.

2.- Eliminacion de una linea.-

Mueve un asterisco a la primera de las 50 posiciones que corresponden a la linea que se desea eliminar.

3.- Activacion de una linea.-

Mueve un blanco que pasa a reemplazar el asterisco que se encuentra en la primera de las 50 posiciones que corresponden a la linea que se desea activar.

4.- Insercion de una linea.-

Chequea que la consulta tenga menos de 10 lineas, blanquea un arreglo de 512 caracteres, a este mueve todas las lineas incluyendo la linea despues de la cual desea insertar, deja un blanco y mueve las 49 caracteres correspondientes a la linea que esta insertando, luego mueve las lineas restantes y finalmente mueve este arreglo de 512 posiciones al arreglo original.

CONSULTA.-

Parametro de entrada: archivo de consultas catalogadas

Parametro de salida: Presentacion en pantalla del archivo.

Realiza 2 tipos de proceso:

- Consulta de una linea.-

Consiste en la localizacion de la linea que desea

consultar y desplegarla.

- Consulta completa.-

Despliega las 50 posiciones correspondientes a cada linea y cuando encuentra una linea que este en blanco termina el proceso.

10. GENERADOR DE REPORTES (POLIREP)

10.1 OBJETIVO

El objetivo principal del Generador de Reportes POLIREP, es facilitar al usuario de POLIREL la utilizacion de la base en lo que respecta a listados o reportes de las relaciones por medio de formatos definidos por el usuario.

Este Generador de Reportes POLIREP presta mucha flexibilidad pues es de facil manejo y util para gran variedad de listados.

10.2 GRAMATICA

10.2.1 NOMBRES:

Los nombres de los archivos de formato son de hasta 4 caracteres, siempre el primer caracter debe ser una letra.

El sistema POLIREP siempre le antepone al nombre del formato las letras FTO asi se evita la confusion con otros archivos de trabajo de la base.

FTOxxxxn.DATA

donde:

xxxx : Representa los 4 caracteres que maximo puede tener el nombre del formato.

n : Representa al # de archivo de formato, pues POLIREP crea para cada formato 3 archivos de data; ese # se lo pone POLIREP.

10.2.2 OPERADORES:

10.2.2.1 OPERADORES DE EJECUCION:

Los operadores que puede utilizar POLIREP a nivel de ejecucion son:

* * Multiplicacion.
* / Division.

- Resta.

+ Suma.

10.2.2.2 OPERADORES DE CRITERIOS DE SELECCION:

Los criterios de seleccion son facilidades que se le da al usuario para seleccionar al momento de la impresion los datos, o sea, que el usuario de acuerdo a la gramatica de los criterios de seleccion puede decidir bajo que criterio se imprime o no una columna. La gramatica que se mantiene para los criterios de seleccion depende de la estrategia de la columna a listarse, asi:

C : Se permite la comparacion con 1 o 2 caracteres usando 'OR'.

S : Se permite la comparacion con 1 o 2 cadenas de caracteres usando 'OR'.

R : Se permite la comparacion de hasta 2 valores reales sea por >, < o =. Esta comparacion puede hacerse por 'OR' o por 'AND'. Los valores deben ser reales de hasta 5 numeros enteros y maximo 2 decimales; el

PUNTO DECIMAL ES OBLIGATORIO, los 2 decimales no son requeridos.

E : Se permite la comparacion de hasta 2 valores enteros sea por >, < o =. Esta comparacion puede hacerse por 'OR' o por 'AND'. Los valores deben ser enteros de hasta 5 caracteres.

10.3 ARCHIVOS

Por cada formato se crean 3 archivos:

1 -> TDTFORMAT

2 -> ADFCABEC

3 -> TFDETALLE

El archivo con etiqueta TDTFORMAT es el archivo descriptor de formato general y cabeceras.

El archivo con etiqueta ADFCABEC es el archivo de datos de cabeceras.

El archivo con etiqueta TFDETALLE es el archivo descriptor de detalle.

10.3.1 ARCHIVO DESCRIPTOR DE FORMATO Y CABECERAS.-

(TDTFORMAT)

Este archivo es de 1 bloque de dimension, y sirve para grabar los datos de formato general y los datos de las lineas de cabeceras.

POSICION	DESCRIPCION	
1 - 1	Tipo de listado	
	(1 = Listado de 1 relacion	
	2 = Listado de 2 relaciones)	
2 - 3	Lineas por pagina	(1...80)
4 - 6	Caracteres por linea	(1..132)
7 - 7	# de lineas de cabecera	(1....7)
8 - 8	# de lineas de detalle	(1...3)

De acuerdo al tipo de listado (1 o 2 relaciones), se graban los datos, si se trata de un listado tipo 1 (listado de 1 relacion) se grabaran los siguientes datos:

9 - 9	Relacion a listarse	
10 - 10	Forma de lectura	
	(N = Normal, lectura secuencial	
	I = a traves de indice invertido	
	C = a traves de archivo Clasific.)	
11 - 12	# de Columna Indice o Clasificacion	
13 - 15	1era. posicion en ADFCABEC	

por forma de lectura I.

16 - 16 # criterios (forma I).

Si se trata de un listado tipo 2 (listado de 2 relaciones), se pediran los siguientes datos:

9 - 9 Relacion principal a listarse

10 - 11 Columna criterio para relacion 2

12 - 12 Relacion # 2

13 - 14 Columna indice en relacion 2

15 - 16 -- libre --

Los siguientes datos se graban para ayuda en la ejecucion del formato:

17 - 17 Si hay quiebre a nivel de detalle (se graba S)

18 - 18 Si hay quiebre por cabeceras (se graba S)

19 - 22 -- libre --

23 - 23 Si se pide fecha (se graba F)

24 - 24 # de bloques archivo descriptor de detalle (TFDETALLE)

25 - 27 Siguiente posicion disponible en ADFCABEC

28 - 29 Numero de columna de quiebre de

cabecera

30 - 30 Nombre de relacion para lectura directa.

31 - 32 # de columna de relacion directa

33 - 40 -- libre --

Las posiciones de la 1 a la 40 se las ha dejado para los datos del formato general.

Desde la posicion 41 a la posicion 460 se graban los posibles 35 campos de cabeceras. Pueden definirse de 1 a 7 lineas de cabecera, con 5 campos como maximo por cada linea. Cada campo ocupa 12 bytes de longitud.

41 - 41 # de linea de cabecera

42 - 42 Tipo de campo de cabecera
(C = Constante
V = Variable
F = Fecha
- = Repeticion)

43 - 43 Salto antes de imprimir (1...5)
(solo en 1er campo de c/linea)

44 - 46 Posicion inicial para imprimir

47 - 49 Offset en ADFCABEC, solo si es

	tipo Constante.
50 - 52	Longitud del valor, solo si es tipo Constante.
53 - 64	Siguiente campo
.. ..	.....
449 - 460	Ultimo campo
461 - 511	-- libre --
512 - 512	Creacion exitosa (*)

(*) El byte 512 sirve para validar que la creacion de los archivos de formato haya sido exitosa, este byte es chequeado en cada programa de POLIREP.

10.3.2 ARCHIVO DE DATOS DE CABECERAS.-

(ADFCABEC)

Este archivo es de 1 bloque de dimension y se usa para grabar los datos de las constantes de las cabeceras y ademas grabar los atributos usados en la lectura por indice invertido, estos atributos se graban de la posicion 1 a la 50 y las constantes de la 51 a la 512.

10.3.3 ARCHIVO DESCRIPTOR DE DETALLE.-

(TFDETALLE)

Este archivo es de varios bloques de dimension y se usan para grabar los datos de las lineas de detalle.

Desde la posicion 1 a la posicion 507 se graban los datos de cada campo de las lineas de detalle, como maximo se permiten 25 campos, 13 campos para el 1er. bloque y 12 campos para el 2do. bloque.

POSICION	DESCRIPCION
1 - 1	# de linea de detalle
2 - 2	Tipo de campo de detalle (N = Normal V = Virtual)
3 - 3	Relacion a listarse, cuando se trabaja con 2 relaciones.
4 - 5	Columna a listarse
6 - 6	Pide constante en ejecucion (si el campo es tipo V)
7 - 8	Columna 2 para campo tipo V
9 - 9	Salto antes imprimir (1...5) (solo en 1er campo de c/linea)
10 - 12	Posicion incial para impresion

13 - 15	Posicion final para impresion
16 - 16	Si se saca promedio por ese campo (solo si estrategia = 'E' o 'R')
17 - 17	Indicador de la forma de listar el campo: (N = Normal Q = Quiebre T = Total, solo si la estrategia es igual a 'E' o 'R')
18 - 18	Operacion a realizarse si el campo es tipo Virtual (*,+,-,/)
19 - 39	Criterio de seleccion
40 - 78	Siguiente campo
.. ..	.....
469 - 507	Ultimo campo del bloque
508 - 512	-- libre --

10.4 CREACION DE FORMATOS

OBJETIVO:

El objetivo principal de este programa es permitir al usuario definir los formatos que el desea dentro de los parametros establecidos.

PROCESOS:

Existen 3 procesos esenciales en la creacion de un formato:

- Creacion de datos de formato general
- Creacion de datos de cabeceras
- Creacion de datos de detalle

DESCRIPCION DEL PROCESO GENERAL:

Este programa pide al inicio el nombre con que se va a grabar el nuevo formato y el # de relaciones que se va a listar. Luego de esto se ejecutan en el orden definido los 3 procesos esenciales.

10.4.1 CREACION DE DATOS DE FORMATO GENERAL

PARAMETROS:

- # de lineas por pagina: Desde 1 hasta maximo 80 lineas por pagina.
- # caracteres por linea: Desde 1 caracter hasta 132 caracteres.
- # lineas de cabeceras: Se pueden definir desde 1 linea hasta 7 lineas de cabecera, cada linea debe tener maximo 5 campos.
- # de lineas de detalle: Desde 1 linea de detalle hasta 3, pueden definirse hasta 10 campos en cada linea.
- Relacion a listarse: Es la relacion que se va a listar, si se lista solo una relacion, esta es la relacion a listarse y si se listan 2, esta es la relacion principal.
- LISTADO DE 1 RELACION:

  Lectura por indice invertido, por archivo de clasificacion o lectura normal (N,I,C): Se pone la inicial de acuerdo al tipo de lectura que se desea.
- LISTADO DE 2 RELACIONES:

  Se debe definir la columna de la relacion principal que servira de atributo para la

relacion 2 que sera leida por indice invertido, y se define ademas la relacion # 2 y la columna de esa relacion que es sobre la cual se va a realizar la lectura directa, se valida que esa columna tenga indice invertido (Ver numeral 9.2).

DESCRIPCION DEL PROCESO:

Se procede a hacer la peticion de cada uno de los parametros definidos anteriormente, y se pasan a grabar en las posiciones definidas en el Numeral 10.3.1.

De cada parametro se hacen las validaciones respectivas, que los valores esten dentro del rango debido, que las relaciones existan, que las columnas existan en la relacion, etc.

Cuando se trata del listado tipo 1 (listado de 1 sola relacion), solo se pide por pantalla la forma en que se va a hacer la lectura de acuerdo al tipo de lectura se van pidiendo los demas parametros.

Cuando se trata de un listado tipo 2 (listado de 2 relaciones) entonces luego de pedido el nombre de la relacion principal se piden los datos de la columna que servira de atributo para la

segunda relacion y la columna en la segunda relacion sobre la cual se va a hacer la lectura directa por indice invertido.

10.4.2 CREACION DE DATOS DE CABECERAS

PARAMETROS:

- # campos de cada linea: Se pide por cada linea y maximo pueden ser 5 campos por cada linea.

En cada campo de cada linea de cabecera se piden los siguientes datos:

- Tipo de cabecera: Se pide los 4 tipos posibles (C,V,F,-), solo 1 vez se pueden definir campos tipo V o F, si se definen 1 vez luego ya no es permitido definir otro campo de ese tipo nuevamente.

  C: Constante, es un campo donde solo se permite definir un valor constante que sera el mismo que vaya en el listado.

  V: Variable, es un campo que implica un quiebre de control a nivel de cabecera, se permite solo un campo de este tipo en todo formato. Se definen ademas si se desea una relacion para lectura directa si se desea sacar algun dato de otra relacion.

  F: Fecha, este campo hace que se pida fecha al inicio de la ejecucion de un formato, solo se permite un campo de este tipo en todo el formato.

  -: Rayado, es una facilidad que se le da al

usuario para adornar las cabeceras de un reporte y lo que hace es llenar la linea de rayas.

Si se definio el campo tipo V, entonces se pediran los siguientes datos:

- Columna a listarse:

  Debe ser una columna valida dentro de la relacion principal.

- Relacion lectura directa:

  Si se define una relacion, se valida si la misma existe.

- Columna a listarse:

  Este campo se pide si se desea hacer lectura directa.

Por cualquier tipo de campo se piden lo siguientes datos:

- Salto antes de imprimir:

  Solo se pide este dato cuando se trata del primer campo de la linea, como maximo puede ponerse el # 5.

- Posicion inicial:

  Se pide la posicion inicial de impresion, como maximo se puede poner el numero de caracteres

por linea definidos en la creacion del formato general.

Cuando el campo es tipo C (constante) se solicita ademas el siguiente dato:

- Valor:

  Es la constante que se desea a nivel de cabecera y su longitud no puede ser mayor que la longitud definida de la linea.

DESCRIPCION DEL PROCESO:

Se pide los parametros definidos anteriormente y se graban de acuerdo a lo definido en el Numeral 10.3.2.

De cada parametro se validan los rangos establecidos y en el caso de una relacion se valida si existe, y si se trata de una columna se valida si existe en la relacion definida.

Cuando se trata del listado de 2 relaciones, se permite la opcion de tipo Variable, mas no se permite poner una relacion para lectura directa por medio de POLIREP.

10.4.3 CREACION DE DATOS DE DETALLE

PARAMETROS:

- # campos de cada linea: Se pide por cada linea y maximo pueden ser 10 campos por cada linea.

Por cada campo de cada linea se piden los siguientes datos:

- Tipo de campo de detalle: Se pide los 2 tipos posibles (N,V).

  N : Normal, es un campo que solo se toma un numero de la columna y se imprime su valor.

  V : Virtual, es un campo que se genera dentro de POLIREP para facilitar al usuario la manipulacion de 2 campos realizando alguna operacion de las ya definidas, el resultado de esa operacion es lo que se imprime en el reporte.

- Columna a listarse:

  Esta columna se valida si existe dentro de la relacion a listarse.

Cuando el campo es tipo V (Virtual), se piden los siguientes datos:

- Desea pedir constante?:

A esta pregunta se debe responder S/N, si la respuesta es N se pide:

- # columna para operacion:

  Se valida que exista dentro de la relacion principal y tenga la misma estrategia que la columna # 1.

- Operacion a realizarse:

  Debe definirse que operacion se desea realizar, sea entre las 2 columnas, o entre la 1era. columna y la constante que se pide a nivel de ejecucion.

- Salto antes de imprimir:

  Solo se pide este dato cuando se trata del primer campo de la linea, como maximo puede ponerse el # 5.

- Posicion (x,y):

  Se pide la posicion inicial y final de impresion, como maximo se puede poner en posicion inicial el # de caracteres por linea menos 1 y la posicion final puede ser el total de caracteres. Ademas se valida que esas posiciones no esten usadas anteriormente, y que la posicion inicial siempre sea menor o igual a la posicion final.

- Desea promedio S/N :

  Si la columna a listarse es de estrategia 'E' o 'R', se pide este dato.

- Quiebre, Total o Normal:

  Siempre que en otro campo de detalle no se haya definido ya la opcion de Quiebre, entonces se puede definir esta opcion. La opcion de Total se puede definir solo si la columna a listarse es de estrategia 'E' o 'R'.

- Criterio de Seleccion:

  Se pide para cada campo y se valida de acuerdo a la gramatica definida en el Numeral 10.2.2.2.

DESCRIPCION DEL PROCESO:

Se piden los parametros definidos anteriormente y se graban de acuerdo al formato definido en el Numeral 10.3.3.

Por cada linea de detalle se pide el # de campos que se va a definir y de acuerdo a ese numero se van pidiendo los datos.

Cuando se trata de un listado de 2 relaciones, se pide el # de campos a definirse por cada relacion, y con este dato se graba la relacion a listarse por cada campo.

Se hacen las validaciones de acuerdo a los rangos de cada dato y ademas se valida: si existe la columna, la sintaxis del criterio de seleccion de acuerdo a la gramatica, etc.

10.5 CONSULTA DE FORMATOS

OBJETIVO:

El objetivo principal de este programa es permitir al usuario consultar los formatos que ha definido anteriormente.

PROCESOS:

Existen 3 procesos esenciales en la consulta de un formato:

- Consulta de datos de formato general
- Consulta de datos de cabeceras
- Consulta de datos de detalle

CONSULTA DEL PROCESO GENERAL:

Este programa pide al inicio el nombre del formato que se va a consultar y presenta una pantalla de las 3 opciones que tiene y que son los procesos definidos anteriormente.

10.5.1 CONSULTA DE DATOS DE FORMATO GENERAL

PARAMETROS:

Los parametros que se usan para la consulta de los datos generales son los mismos que se definieron en la creacion. (Ver Numeral 10.4.1).

DESCRIPCION DEL PROCESO:

Se presenta en pantalla los datos del formato general sacados del TDTFORMAT de las posiciones definidas en el Numeral 10.3.1. Se presentan los campos de acuerdo a los diferentes casos, si es listado tipo 1 (listado de 1 sola relacion), se sacan los datos referentes a este tipo de listado y si se trata de un listado tipo 2 (listado de 2 relaciones), se sacan los datos de este tipo.

10.5.2. CONSULTA DE DATOS DE CABECERAS

PARAMETROS:

Parametros de entrada:

- # linea a consultar:

  Se pide el # de linea de cabecera a consultar.

- # campo a consultar:

  Tambien se pide el # del campo de esa linea que se desea consultar.

Parametros de salida:

Los datos que se presentan son los mismos que se solicitaron al momento de la creacion de las cabeceras. (Ver Numeral 10.4.2).

DESCRIPCION DEL PROCESO:

Se pide el # de la linea a consultar y el # de campo y se calcula la posicion dentro del buffer que ocupa ese campo.

Se presentan los datos solicitados en una pantalla, estos datos son sacados del TDTFORMAT y del ADFCABEC de acuerdo a como fueron definidos en el Numeral 10.3.2.

10.5.3 CONSULTA DE DATOS DE DETALLE

PARAMETROS:

Parametros de entrada:

- # de linea a consultar:

  Se pide el # de la linea de detalle que se desea consultar.

- # de campo a consultar:

  Se pide el # del campo dentro de la linea de detalle que se va a consultar.

Parametros de salida:

Los datos que se presentan son los mismos que se piden en el momento de la creacion de los datos de detalle (Ver Numeral 10.4.3).

DESCRIPCION DEL PROCESO:

Se pide el # de la linea de detalle a consultar y el # de campo y con estos datos se calcula el # de bloque y la posicion dentro del bloque de ese campo.

Los datos presentados se sacan del TFDETALLE y son tal cual estan grabados de acuerdo al Numeral 10.3.3.

10.6 CORRECCION DE FORMATOS

OBJETIVO:

El objetivo principal de este programa es permitir al usuario corregir los formatos que el ha definido anteriormente.

PROCESOS:

Existen 3 procesos esenciales en la correccion de un formato:

- Correccion de datos de formato general
- Correccion de datos de cabeceras
- Correccion de datos de detalle

CORRECCION DEL PROCESO GENERAL:

Este programa pide al inicio el nombre del formato que se va a corregir y presenta una pantalla de las 3 opciones que tiene y que son los procesos definidos anteriormente.

Luego pide cual de las 3 opciones desea y le permite la correccion de acuerdo a la opcion elegida.

10.6.1 CORRECCION DE DATOS DE FORMATO GENERAL

PARAMETROS:

Los parametros que se usan para la correccion de los datos generales son los mismos que se definieron en la creacion. (Ver Numeral 10.4.1).

DESCRIPCION DEL PROCESO:

Se presenta en pantalla los datos del formato general sacados del TDTFORMAT de las posiciones definidas en el Numeral 10.3.1. Se presentan los campos de acuerdo a los diferentes casos, si es listado tipo 1 (listado de 1 sola relacion), se sacan los datos referentes a este tipo de listado y si se trata de un listado tipo 2 (listado de 2 relaciones), se sacan los datos de este tipo.

Se permite la correccion de todos los datos que se presentan numerados en la pantalla, y se hacen las mismas validaciones que se hicieron al crear el formato.

10.6.2 CORRECCION DE DATOS DE CABECERAS

PARAMETROS:

Parametros de entrada:

- # linea a corregir:

  Se pide el # de linea de cabecera a corregir.

- # campo a corregir:

  Tambien se pide el # del campo de esa linea que se desea corregir.

Parametros de salida:

Los datos que se presentan son los mismos que se solicitaron al momento de la creacion de las cabeceras. (Ver Numeral 10.4.2).

DESCRIPCION DEL PROCESO:

Se pide el # de la linea a corregir y el # de campo y se calcula la posicion dentro del buffer que ocupa ese campo.

Se presentan los datos solicitados en una pantalla, estos datos son sacados del TDTFORMAT y del ADFCABEC de acuerdo a como fueron definidos en el Numeral 10.3.2.

Se pide luego cual es el dato a cambiar de los que se hallan en pantalla y se recibe el dato y se graba en la misma posicion en que estaba

grabado el anterior y haciendose las mismas validaciones que al crear el formato.

10.6.3 CORRECCION DE DATOS DE DETALLE

PARAMETROS:

Parametros de entrada:

- # de linea a corregir:

  Se pide el # de la linea de detalle que se desea corregir.

- # de campo a corregir:

  Se pide el # del campo dentro de la linea de detalle que se va a corregir.

Parametros de salida:

Los datos que se presentan son los mismos que se piden en el momento de la creacion de los datos de detalle (Ver Numeral 10.4.3).

DESCRIPCION DEL PROCESO:

Se pide el # de la linea de detalle a corregir y el # de campo y con estos datos se calcula el # de bloque y la posicion dentro del bloque de ese campo.

Los datos presentados se sacan del TFDETALLE y son tal cual estan grabados de acuerdo al Numeral 10.3.3.

Luego se pide cual es el dato a corregir y se lo graba en la misma posicion donde se hallaba el

anterior. Las validaciones de cada dato son las mismas que se hacen al momento de la creacion.

10.7 ELIMINACION DE FORMATOS

OBJETIVO:

El objetivo principal de este programa es permitir al usuario eliminar los formatos que el ha definido anteriormente.

PARAMETROS:

Los parametros que se usan para la eliminacion de un formato son la presentacion de los datos generales para que el usuario se asegure de que es verdaderamente el formato que desea eliminar. Los datos del formato general son los mismos que se definieron en la creacion. (Ver Numeral 10.4.1).

DESCRIPCION DEL PROCESO:

Se presenta en pantalla los datos del formato general sacados del TDTFORMAT de las posiciones definidas en el Numeral 10.3.1. Se presentan los campos de acuerdo a los diferentes casos, si es listado tipo 1 (listado de 1 sola relacion), se sacan los datos referentes a este tipo de listado y si se trata de un listado tipo 2 (listado de 2 relaciones), se sacan los datos de este tipo.

Luego se consulta si se desea eliminar el formato presentado en pantalla y si es asi se eliminan los 3 archivos de ese formato del disco.

10.8 EJECUCION DE FORMATOS

OBJETIVO:

El objetivo principal de este programa es permitir al usuario ejecutar los formatos que el desea dentro de los parametros establecidos.

10.8.1 ESTRUCTURAS DINAMICAS

Para poder efectuar la ejecucion de un formato se preciso de 3 estructuras dinamicas en memoria que son generadas con los datos grabados en los 3 archivos de formato.

La lista enlazada que corresponde a las cabeceras guarda los siguientes datos, tomados del TDTFORMAT y del ADFCABEC, somo se definio en capitulos anteriores.

- # de linea de cabecera
- Tipo de linea, para saber la opcion a seguir de acuerdo a cada tipo.
- Salto antes de imprimir, que es el que se va a usar antes de imprimir una linea.
- Posicion dentro de la linea
- Valor de cabecera, cuando se trata de un campo tipo Constante.
- Longitud de la constante, que se usa para sacar del ADFCABEC el valor.
- Siguiente nodo, apuntador al siguiente nodo de la lista enlazada de cabeceras.

La lista enlazada que corresponde a los datos de

detalle guarda los siguentes datos tomados del TFDETALLE como se definio en numerales anteriores.

- # de linea de detalle
- Relacion a listarse, este dato se usa cuando se trata de un listado tipo 2.
- Tipo de campo de detalle, sirve para saber que opcion se debe realizar de acuerdo al tipo de campo.
- # columna 1, # de la columna que se va a imprimir, o si el campo es Virtual, el # de la columna para realizar la operacion.
- # columna 2, segundo # de columna para operacion.
- Estrategia, al momento de ejecutarse se carga la estrategia de cada campo.
- Pedir constante, en este caracter se pone si es que se debe pedir constante al momento de ejecucion para este campo.
- Operacion, aqui se pone la operacion a realizarse cuando el campo es Virtual.
- Salto, sirve para poner el salto que se desea antes de imprimir una linea.
- Posicion1, sirve para determinar la posicion

inicial de impresion en la linea de detalle.

- Posicion2, sirve para determinar la posicion final de impresion en la linea de detalle.
- Promedio, caracter donde se graba si se saca promedio por ese campo (S/N).
- QST, en este campo se graba el indicador de forma de listado de este campo (Quiebre, Total, Normal).
- Criterio, aqui ponemos el criterio de seleccion para cada campo.
- Constante, una vez que se ha pedido la constante para campo Virtual se le graba en esta posicion para que no se tenga que volver a pedir.
- Pointotales, apuntador a la estructura dinamica correspondiente a los totales; para cada campo de detalle que se requiera la opcion 'T' de total, se genera un nodo de la estructura de totales.
- Siguiente nodo, apuntador al siguiente nodo de los campos de detalle.

Los datos de los nodos de la estructura dinamica de totales tiene como campos:

- IPromedio (C):

Campo donde se acumulan los valores enteros para sacar promedio, acumulador de promedio por quiebre de cabeceras.

- ISubtotal (C):

  Campo donde se acumulan los valores enteros para sacar subtotales por quiebre de cabeceras.

- ITotal (C):

  Campo donde se acumulan los valores enteros para sacar total por quiebre de cabeceras.

- RPromedio (C):

  Campo donde se acumulan los valores reales para sacar promedio, acumulador de promedio por quiebre de cabeceras.

- RSubtotal (C):

  Campo donde se acumulan los valores reales para sacar subtotales por quiebre de cabeceras.

- RTotal (C):

  Campo donde se acumulan los valores reales para sacar totales por quiebre de cabeceras.

- #total (C):

  Campo donde se acumula la cantidad de numeros para sacar promedio, acumulador por quiebre de cabeceras.

- IPromedio (D):

  Campo donde se acumulan los valores enteros para sacar promedio, acumulador de promedio por quiebre de detalle.

- ISubtotal (D):

  Campo donde se acumulan los valores enteros para sacar subtotales por quiebre de detalle.

- RPromedio (D):

  Campo donde se acumulan los valores reales para sacar promedio, acumulador de promedio por quiebre de detalle.

- RSubtotal (D):

  Campo donde se acumulan los valores reales para sacar subtotal por quiebre de detalle.

- #total (D):

  Campo donde se acumula la cantidad de numeros para sacar promedio, acumulador por quiebre de detalle.

- Pointdetalle :

  Apuntador al nodo correspondiente de detalle.

- Signodo:

  Apuntador al nodo siguiente de la lista enlazada de totales.

10.8.2 IMPRESION DE CABECERAS

Para imprimir las lineas de cabeceras se barre la lista enlazada de datos de cabecera y de acuerdo a cada tipo de campo se mueven los datos.

Cuando se cambia de linea entonces se imprime la linea anterior.

Si el campo es CONSTANTE se mueve el valor constante desde la posicion inicial, el # de caracteres igual a la longitud de la constante.

Si el campo es FECHA se mueve la fecha que ha sido pedida al inicio de la ejecucion, desde la posicion inicial.

Si el campo es "-" se mueve tantas rayitas como caracteres por linea se hayan definido.

Si el campo es VARIABLE entonces se toma el valor de la columna1 que se ha definido; si se definio una relacion para lectura directa con el valor de la columna1 se hace un RETIRODATO de esa relacion y se saca el valor de la columna2, que es el que se mueve a la linea de impresion desde la posicion inicial.

10.8.3 IMPRESION DE DETALLE

10.8.3.1 LISTADO DE 1 RELACION

Cuando se trata del listado de 1 relacion, luego de efectuada la lectura secuencial (LECSECTU), o la lectura directa (LECDIRTU), y luego de sacadas las lineas de cabeceras, se procede a la impresion de las lineas de detalle.

10.8.3.2 LISTADO DE 2 RELACIONES

Cuando se trata del listado de 2 relaciones, luego de efectuada la lectura secuencial de la primera relacion (LECSECTU) y luego de la impresion de las lineas de cabeceras, se procede a imprimir los campos de las lineas de detalle que sean de la relacion1; una vez que se han impreso todos los campos de esa relacion entonces se lee basandose en la columna de criterio, se leen por medio de la lectura directa (LECDIRTU) todas las tuplas que satisfagan el criterio de la columna dada, y se imprimen luego

lineas por c/tupla hasta que se acaben las tuplas por ese criterio y luego se lee la siguiente tupla con (LECSECTU).

DESCRIPCION DEL PROCESO

De acuerdo al tipo de listado se sabe cual es el 1er. campo a imprimir, y de acuerdo al tipo de campo se realiza el movimiento de los datos:

N : Normal,cuando el campo es normal solo se mueven los datos de acuerdo a la columna y estrategia; desde la posicion1 a la posicion2. Si la longitud del dato es mayor se trunca la diferencia y solo se mueven entre esas posiciones.

V : Virtual, cuando el campo es virtual se toma el dato de la columna1 y si se pide constante se efectua la operacion con la constante, y si no se saca el valor de la columna2 y se efectua la operacion con la columna2. Se pide la constante para ese campo solo la primera vez, luego se toma el valor de la estructura dinamica.

Una vez que ya se tiene el valor a imprimir se efectua la seleccion del campo de acuerdo al criterio de seleccion. Si el criterio no se

satisface entonces se lee la siguiente tupla y se vuelve a decidir cual es la opcion a seguir y se mueve desde el campo correspondiente de acuerdo al tipo de listado.

Si el criterio si se satisface o no existe ningun criterio a elegir se pregunta si al campo se le va a sacar promedio, y si es asi se acumula su valor de acuerdo a la estrategia.

Si en la opcion QST se ha puesto sea Q, o T se efectua de acuerdo a eso lo siguiente:

Q : Quiebre, si se da esta opcion solo se verifica si el valor anterior era igual a este para no imprimirlo.

T : Total, si se da esta opcion se acumulan los valores de acuerdo a la estrategia.

Luego de esto se imprime el valor del campo.

10.8.4 CONTROL DE QUIEBRES

El control de los 2 quiebres que permite POLIREP son 1 quiebre a nivel de cabeceras y 1 quiebre a nivel de detalle.

El control de quiebres se efectua luego de la lectura del archivo, se guarda el # de la columna de quiebre de cabecera y se guarda el # de la columna de quiebre de detalle.

Se obtiene el valor de la columna de quiebre de cabecera si es que existe este quiebre, y si no es igual al valor anterior se ejecuta la opcion de impresion de totales.

Se obtiene el valor de la columna de quiebre de detalle si es que existe este quiebre, y si no es igual al valor anterior se ejecuta la opcion de impresion de totales.

10.8.5 IMPRESION DE TOTALES

La impresion de totales se efectua de acuerdo a un indicador que se lo pasa el proceso de quiebre cuando se haya un quiebre. Tambien el proceso de impresion de detalle llama a la impresion de totales cuando detecta fin de pagina.

De acuerdo al valor del indicador que le envian este proceso efectua las siguientes opciones:

1 : Este indicador se lo envia el control de quiebres cuando se encontro quiebre a nivel de cabeceras. En este momento se verifica si se definio quiebre a nivel de detalle, si es asi se saca promedio por detalle (si se ha definido) y luego totales por quiebre de detalle, luego promedio por quiebre de cabeceras y subtotales por cabecera. Ademas se manda a impresion de cabeceras.

2 : Este # indica que se ha encontrado quiebre a nivel de detalle (si es que ha sido definido) y subtotales por quiebre de detalle.

3 : Este indicador se lo transfiere la impresion de detalle por fin de pagina y saca totales

solo si no se ha definido quiebres. Si es asi se saca promedios (si se ha definido) y totales por fin de pagina.

4 : Este indicador se pone al final del programa y saca promedios y totales por quiebre de detalle (si se ha definido); promedios y totales por quiebre de cabecera (si se ha definido) y totales generales.

10. CONCLUSIONES

El Sistema de Base de Datos Polirel, llamado Primera Version por los estudiantes que iniciaron este proyecto, sirvio de punto de partida para desarrollar la Version 2.0, la cual es producto de la experiencia dejada por la Version anterior y de nuestra investigacion creativa para mejorarla y complementarla, manteniendo su ideologia basica de ser un producto netamente didactico.

En ella hemos puesto en practica todos aquellos conceptos y estructuras de datos conocidos a traves de nuestros anos de estudio, materializando topicos teoricos adquiridos en la catedra de Estructura de Datos y Sistemas de Computo. Por ejemplo: el Heap de almacenamiento en disco y en memoria, estructuras dinamicas, proceso multiple de archivos, indices invertidos y de clasificacion, manipulacion de buffers de memoria, etc.

El implementar practicamente todos estos conocimientos tuvo ciertas restricciones tanto de Hardware como de Software, teniendo que agotar casi todos los recursos disponibles en el lenguaje PASCAL del APPLE II. Una de estas restricciones preciso de nosotros la adecuada administracion tanto de memoria principal como auxiliar.

A pesar de estos inconvenientes, se ha logrado un producto de caracteristicas aceptables que junto con la interaccion del usuario permitira el desarrollo de cualquier tipo de aplicaciones.

Al finalizar este proyecto nos queda la satisfaccion de haber participado en la realizacion de un sistema de la magnitud y complejidad de la Base de Datos Polirel, que por sus caracteristicas merece ser considerado como un producto sui-generis y de grandes perspectivas en el campo didactico y por que no decirlo, en el area comercial.

12. APENDICES

A) TABLA DE ERRORES

1 ** FALTA NOMBRE DE ESTRUCTURA/COMANDO **

2 ** NOMBRE DE ESTRUCTURA ESPERADO **

3 ** NOMBRE DE ESTRUCTURA INCORRECTO **

4 ** := ESPERADO **

5 ** OPERADOR (:,*,%) ESPERADO **

6 ** NOMBRE DE COLUMNA INCORRECTO **

7 ** OPERADOR RELACIONAL (>,<,=) ESPERADO **

8 ** NOMBRE DE COLUMNA ESPERADO **

9 ** ARGUMENTO PARA SELECCION ESPERADO **

10 ** COLUMNA(S) PARA PROYECCION INCORRECTA(S) **

11 ** EXPRESION INCORRECTA **

12 ** NOMBRE ESTRUCTURA NO PERMITIDA EN SELECCION **

13 ** ESTRUCTURA PERMANENTE INCORRECTA **

14 ** DELIMITADOR (/) ESPERADO **

20 ** NO EXISTE COLUMNA EN RELACION **

21 ** ESTRATEGIA NO VALIDA **

22 ** NOMBRE PERMANENTE EN PROYECCION NO VALIDO **

23 ** VARIABLE PERMANENTE EN PROYECCION NO EXISTE **

24 ** VARIABLE PERMANENTE EN LIST/DISP NO EXISTE **

25 ** VARIABLE PERMANENTE DUPLICADA **

26 ** NOMBRE PERMANENTE EN SELECCION NO EXISTE **

27 ** OPERADOR #1 DE JOIN NO VALIDO/NO EXISTE **

28 ** OPERADOR #2 DE JOIN NO VALIDO/NO EXISTE **

29 ** ESTRUCTURAS FUENTES JOIN INDENTICAS **

30 ** NO EXISTE COLUMNA COMUN EN JOIN **

31 ** HAY MAS DE UNA COLUMNA COMUN EN JOIN **

32 ** NOMBRE PERMANENTE DE JOIN NO EXISTE **

33 ** OPERACION PARA JOIN NO PERMITIDA **

34 ** NOMBRE PERMANENTE DE JOIN NO VALIDO **

35 ** NO EXISTE SELECCION EN TABLA DE SELECCIONES **

36 ** NO EXISTE JOIN EN TABLA DE JOINS **

37 ** NO EXISTE PROYECCION EN TABLA DE PROYECCIONES **

38 ** NUMERO ENTERO EXCEDE A 5 DIGITOS **

39 ** NUMERO REAL INCORRECTO **

40 ** TABLA GENERAL DE PROCESOS LLENA **

41 ** NO EXISTE CONSULTA CATALOGADA **

42 ** COLUMNAS REPETIDAS EN PROYECCION **

43 ** NO EXISTE TUPLAS EN JOIN **

44 ** NO EXISTE TUPLAS EN SELECION **

45 ** TABLA DE PROYECCIONES LLENA **

50 ** RELACION NO TIENE TUPLAS **

51 ** RELACION NO TIENE COLUMNAS **

75 ** RELACION ESTA ELIMINADA **

76 ** RELACION NO ESTA DEFINIDA **

91 ** UNO DE LOS ARCHIVOS NO EXISTE **

94 ** NO EXISTE NOMBRE DE COLUMNA **

99 ** FIN DE ARCHIVOS **

```
101 ** NUMERO DE ARCHIVOS EXCEDE AL MAXIMO **
```

B) TABLA DE CONDICION DE OPERACION

En este apendice se describe el significado de los posibles valores que puede tomar la variable OPCONDICION al utilizar los procedimientos de la unidad Polirel.

VALOR	SIGNIFICADO
00	Operacion exitosa
21	Valor identificador de la clave perdido en el procedimiento LECDIRTU.
22	Numero de columna dado para nombre de archivo no valido (LECDIRTU).
26	No existe archivo invertido para dicha relacion o no encuentra informacion de control en el archivo ADT.
27	No existe tuplas para dicho criterio de seleccion.
50	Numero de tuplas en relacion igual a cero.
51	Numero de columnas en relacion igual a cero.
60	Longitud de atributo a insertar es igual a cero.
70	Numero de registro a eliminar no esta dentro del rango permitido.
71	Invoca al procedimiento ELIMINTU sin

haber leido una tupla previamente.

75 Relacion eliminada.

76 La relacion ha sido solo formateada y no esta definida.

80 Numero de registro a actualizar no esta dentro del rango permitido.

81 Trata de actualizar una tupla sin haber leido una previamente.

82 Atributo tipo string tiene longitud cero.

91 Archivo no existe o un error ha ocurrido al tratar de accesarlo.

92 Nombre de relacion no permitido por Polirel

94 Uno de los atributos en el registro definido por el usuario no valido.

95 Valor de parametro igual a cero en procedimiento para encontrar la tupla fisica.

99 Fin de datos en la lectura secuencial de una relacion (LECSECTU).

100 Fin de datos para archivos invertidos y/o de clasificacion.

101 Trata de leer mas de tres archivos concurrentemente (inicialice el sistema)

102 Error al tratar de accesar al archivo de clasificacion o invertido; el archivo no

ha sido encontrado.

C) PALABRAS RESERVADAS POR LA UNIDAD POLIREL

En este apendice se describem todas aquellas palabras que son utilizadas por la Unidad Polirel por lo tanto tienen el caracter de reservadas y no deben ser utilizadas en los programas del usuario para evitarse resultados inesperados.

CONSTANTES:

cero

maxcolumnas

tuplelength

vacio

zero

TIPOS:

formato

setofchar

tbuffers

tarchivos

VARIABLES:

adf	numadfbuffes
adfbuffer	numcolumnas
adfcounter	numtfbuffers
alfabeto	numeros
bp01	numtuplas
bp02	opcondicion
bp03	pc02
bp04	ppc
capol	pradf
countadf	prtdf
counttf	prtf
dtfcounter	registro
enpol	repol
findata	stadic
index	stpol
indexbuffer	tdf
longtupla	tdfbuffer
mfiles	tf
modos	tfbuffer
nextbucked	tfcounter
nextposic	tuplalogica
nextrecord	tuplanumero

PROCEDIMIENTOS:

actualtu:	Actualiza tuplas.
adiciotu:	Adiciona tuplas.
datocomun:	Obtiene la informacion general sobre una relacion.
elimintu:	Elimina tuplas.
enterostring:	Convierte un entero a string.
finproceso:	Finaliza una operacion sobre una relacion dada.
inicializacion:	Inicializa variables y buffers de trabajo.
lecdirtu:	Lee en forma directa una tupla.
lecsectu:	Lee en forma secuencial las tuplas.
llenoblancos:	Inicializa una variable string con N blancos.
multifile:	Permite la manipulacion concurrentemente de hasta 3 relaciones.
readfbuffer:	Lee un bloque del archivo ADA.
readtfbuffer:	Lee un bloque del archivo AT.
readfisic:	Obtiene los datos de una tupla fisica dada.
realstring:	Convierte un numero real a string.
retirodato:	Obtiene el dato para una columna especifica.
retirocolumna:	Obtiene los datos de una columna.

stringentero: Convierte un string a su correspondiente valor entero.

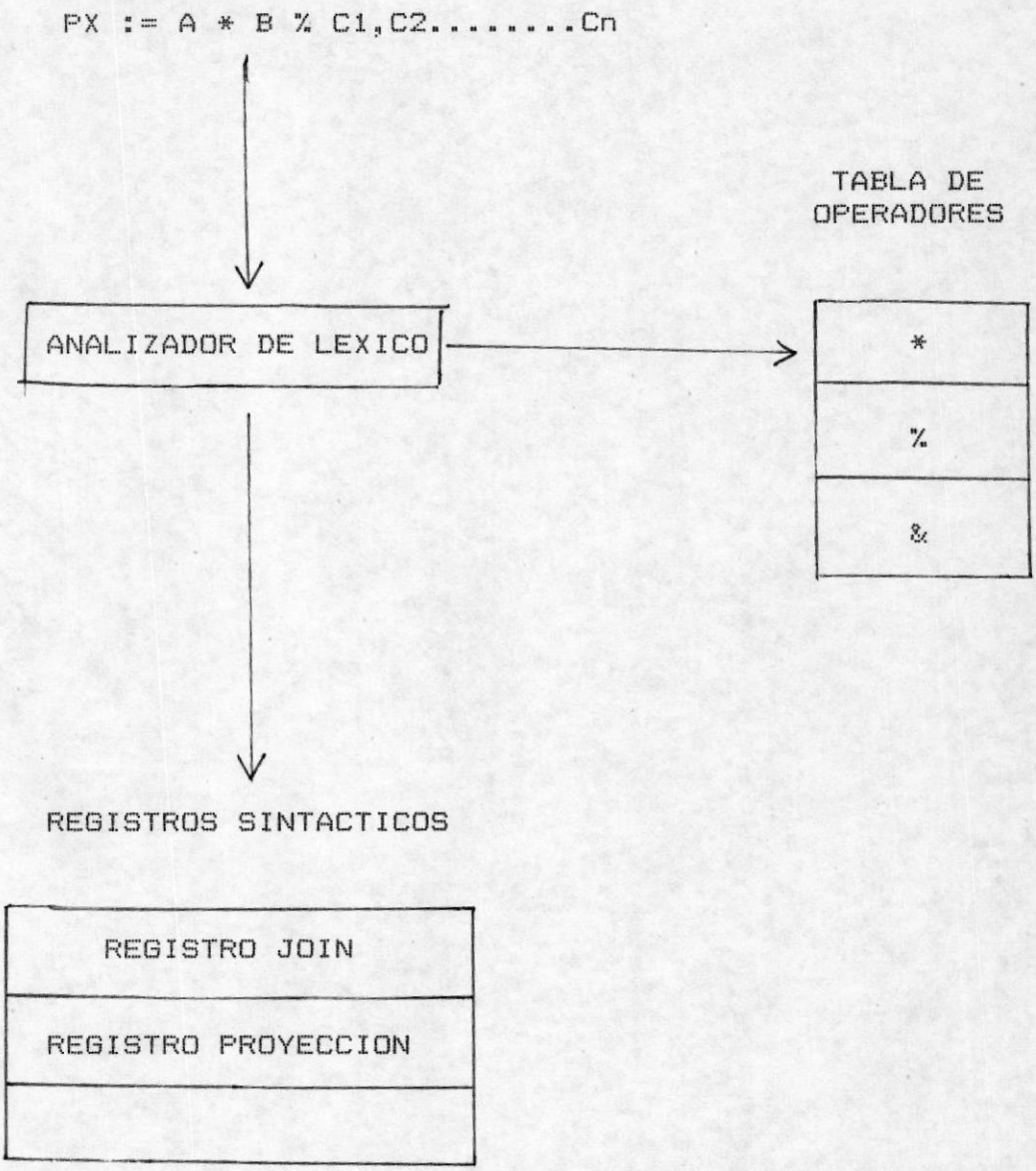
stringreal: Convierte un string a su correspondiente valor real.

tuplafisica: Obtiene el numero de tupla fisica dada una tupla logica.

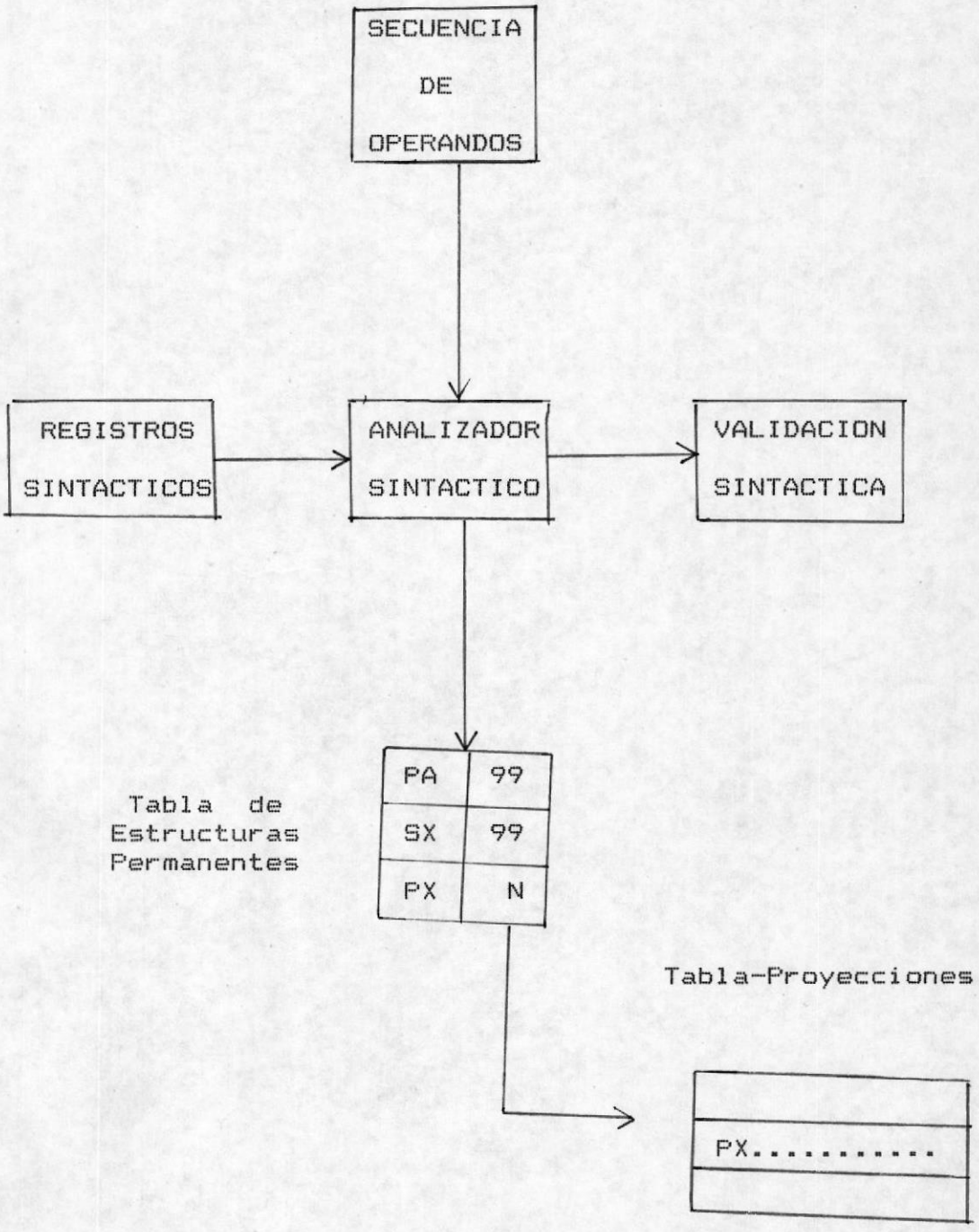
validarchivos: Verifica que existan los archivos de una relacion.

validname: Valida el nombre de una relacion.

D) ANALIZADOR DE LEXICO

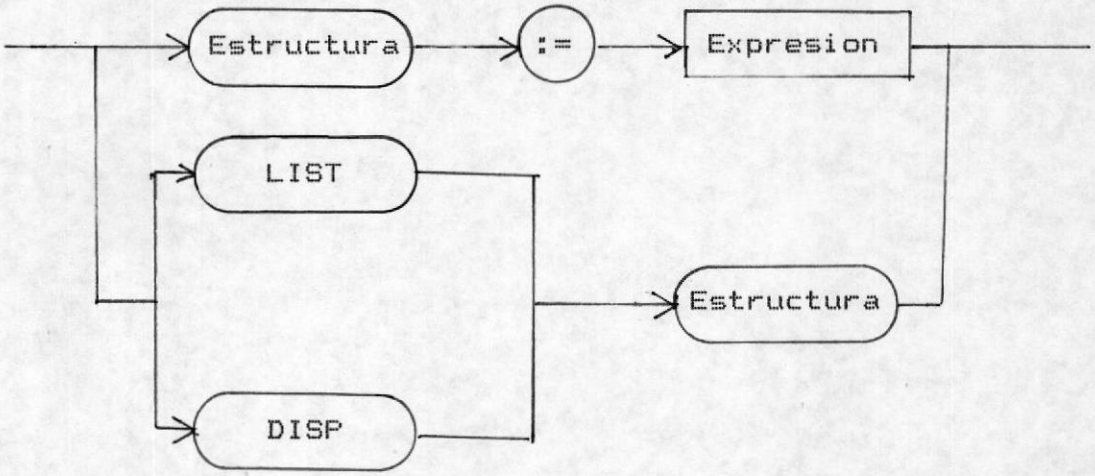


E) ANALIZADOR DE SINTAXIS

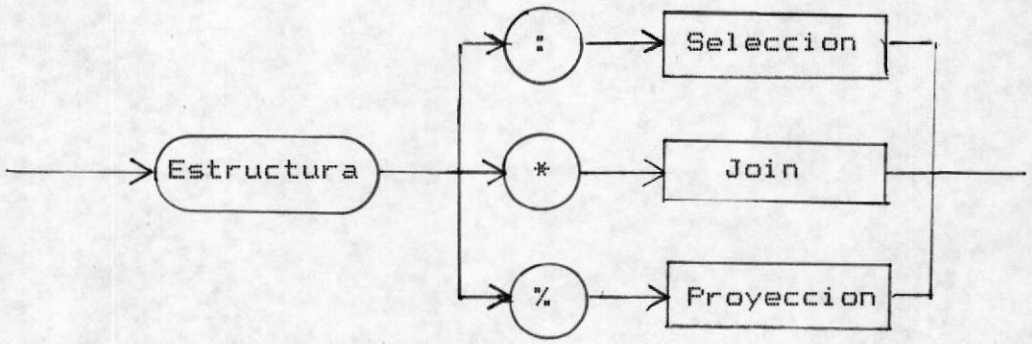


F) DIAGRAMAS SINTACTICOS

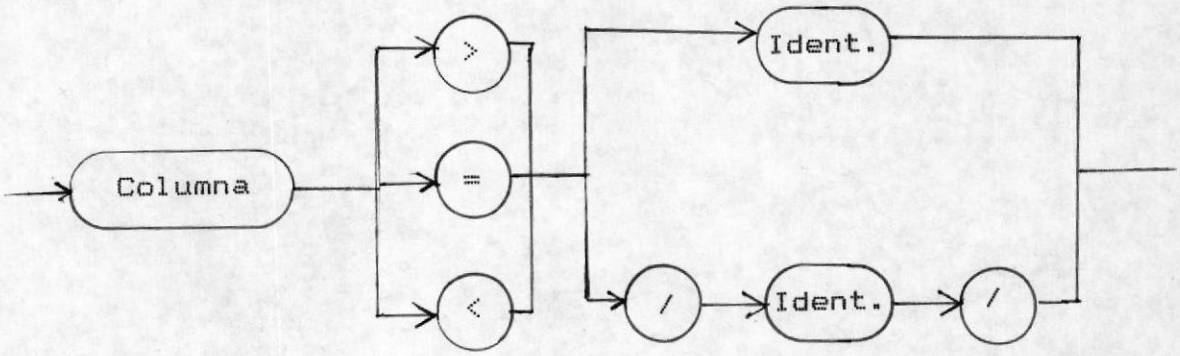
Sentencia



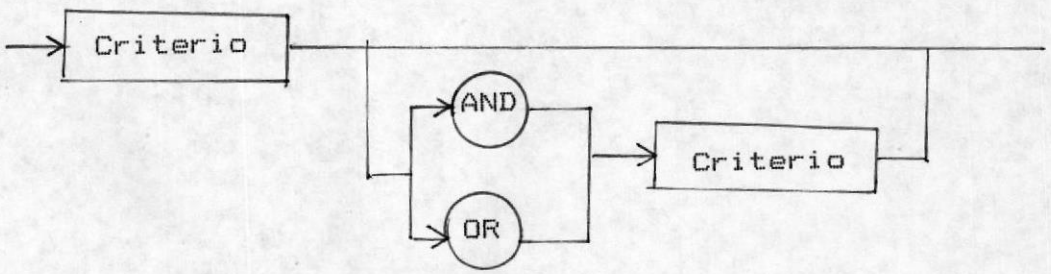
Expresion



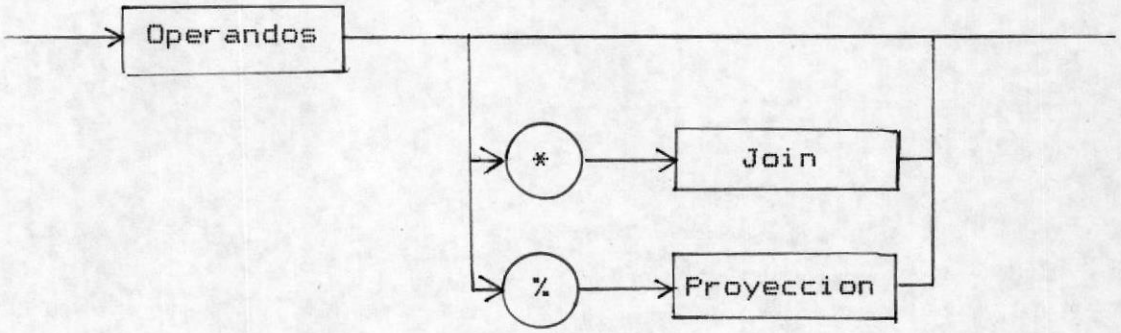
Criterio



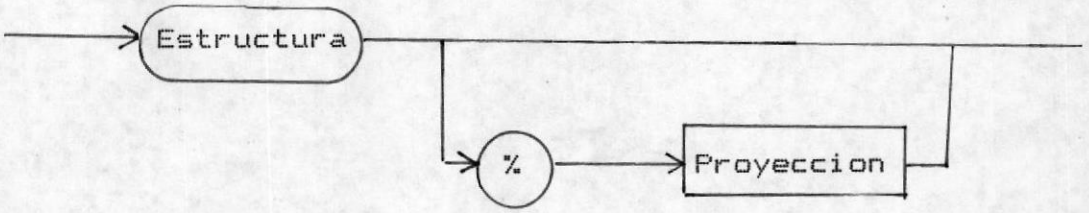
Operandos



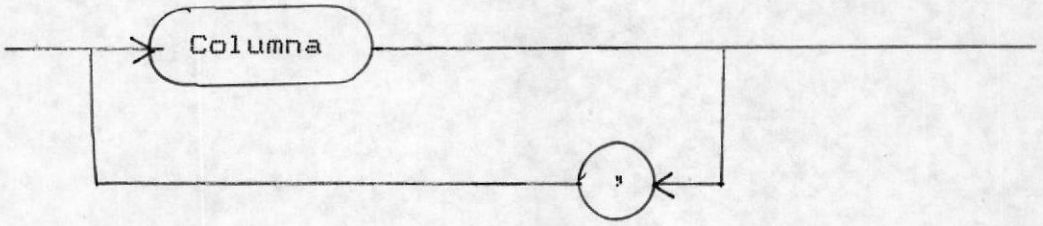
Seleccion



Join



Proyeccion



Identificador