UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

U. A. C. P. P. DEL C. C. H.

SISTEMA DE INFORMACION GENERAL DE ESTRUCTURA RECONFIGURABLE

T E S I S

PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS DE LA COMPUTACION

P R E S E N T A.

VICTOR GERMAN SANCHEZ ARIAS.

MEXICO, D.F. SEPTIEMBRE DE 1980.

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

JURADO:

Presidente:	Dr. Adolfo Guzmán Arenas
Vocal:	Dr. Renato Barrera
Secretario:	Dr. Enrique Calderón
1er suplente:	Dr. Victor Guerra
2do. suplente:	M. en C. Raymundo Segovio

Sitio donde se desarrolló el tema:

Centro de Servicios de Cómputo, UNAM

M. en C. Victor Germán Sánchez Arias

Sustentante

Dr. Adolfo Guzmán Arenas

Asesor

INDICE

I. OBJETIVO

El objetivo de este trabajo se puede dividir en dos partes. Primero, el desarrollo de una estructura básica de un sistema general de información, que permita a través de comandos básicos, realizar todo tipo de consultas, actualizaciones y modificaciones a los datos de una base. Segundo, la implementación al sistema de información, de un mecanismo de optimización de recursos de procesamiento (memoria utilizada y tiempo de respuesta) de manera que su organización se adapte a la forma en que se opera la base de datos a través del tiempo.

I.1 El problema fundamental.

En un sistema como el antes mencionado, donde la base de datos se almacena en dispositivos de acceso directo, el problema fundamental está, en la diferencia que hay entre la organización lógica y la organización física de los datos, de tal suerte que la rapidez de respuesta y cantidad de memoria utilizada en una operación lógica depende de la organización física particular usada (la organización física depende del dispositivo de memoria en particular usado).

El problema puede describirse, de una manera sencilla, de la siguiente forma; supóngase un archivo lógico que puede ser representado por una matriz, como se muestra en la siguiente figura:

	nombre 1	sexo 2	edad 3	---	m
1	JUAN	M	27		
2	MARIA	F	25		
3	ROSA	F	23		
⋮					
n					

columnas – propiedades (características)
renglones – registros

matriz AL n×m

FIG. I.1.A

Los renglones representan los registros y las columnas representan las características con que cuentan cada uno de los registros.

De tal manera que cuando se desee obtener todas las características de un registro X, desde un punto de vista lógico, se puede expresar como:

$$AL(x,j) \quad , j = 1,2,\ldots m \quad (1)$$

de igual forma si se desea conocer el valor de una característica y para todos los registros, puede ser expresado como:

$$AL(i,y) \quad , i = 1,2,\ldots n \quad (2)$$

A través de estas dos funciones básicas de acceso de los datos de un archivo, pueden realizarse operaciones tales como consulta, actualizaciones (bajas, altas, modificaciones), operaciones combinadas, etc. Por ejemplo, si se desea saber quienes son hombres, en el archivo del ejemplo mostrado arriba, resulta muy útil el acceso dado por la función (2), o si se desea conocer todas las características de Juan con el objeto de realizar alguna operación sobre sus valores, la función (1), de acceso, resulta muy adecuada.

Almacenar la matriz AL en una memoria de acceso aleatorio (tal como la memoria principal de una computadora) toma solamente el número de accesos necesarios para consultar una propiedad. Por ejemplo, para obtener el valor AL(3,2),

solo hay que hacer un acceso a la matriz; no hay necesidad de acceso a otros elementos de AL.

Sin embargo, estas dos funciones de acceso no son fácilmente implementadas en una memoria de acceso directo (por ejemplo, disco). La organización de la información lógica es matricial pero su almacenamiento (en este tipo de memorias) siempre será en una dimensión.

La unidad de almacenamiento de un dispositivo de tal naturaleza son los registros físicos, de tal manera, que un registro se almacena a través de esta unidad y solo a través de ella misma puede ser obtenida. Si los registros físicos son asociados a los renglones, sólo se pueden accesar renglones, si se quieren accesar columnas será necesario accesar todos los renglones para formar una columna.

De este problema se deriva la complejidad requerida en la organización física (estructura) y a su vez la necesidad de crear sistemas dinámicos que cambian de estructura para adecuarse a las necesidades del sistema.

Otras alternativas de solución al problema, no estudiadas en este trabajo son: el desarrollo de un "disco" con posibilidad de acceso a renglones y columnas de manera directa, o el uso de memorias aleatorias como medio de almacenamiento, alternativa muy viable, por el contínuo decrecimiento en precio en este tipo de memorias tales como las burbujas magnéticas.

II. JUSTIFICACION DE UN SISTEMA GENERAL RECONFIGURABLE.

Las características básicas del sistema desarrollado son: su generalidad (no orientado a un sistema de información en particular) y su capacidad de adaptación al tipo de operación más común (reconfigurable).

En este capítulo se justificarán ambas características.

II.1 Sistema General de Información.

Una alternativa para manejar una base de datos son los sistemas orientados a una base de datos en particular. Este tipo de sistemas se pueden justificar para grandes bases de datos y para aplicaciones muy específicas.

Un sistema de esta naturaleza con su estructura orientada a la aplicación, significa una estructura óptima en cuanto al manejo de los recursos de la máquina y al tiempo de respuesta requerido por el usuario.

Sin embargo, en sistemas medianos y muy comunes, puede no ser muy justificable que el sistema sea orientado a problemas específicos, pues representa duplicación de esfuerzos al implementar siempre problemas semejantes.

Dentro de las restricciones que generalmente implican los sistemas orientados (haciendo a un lado la ventaja de implementación óptima), están: dado que el sistema está muy orientado a un tipo de aplicación, es difícil extenderlo a otros tipos no previstos en el planteamiento original, además, generalmente los programas de estos sistemas son muy dependientes de la máquina en particular donde se implementó, por lo que es muy difícil su transportación a otras máquinas.

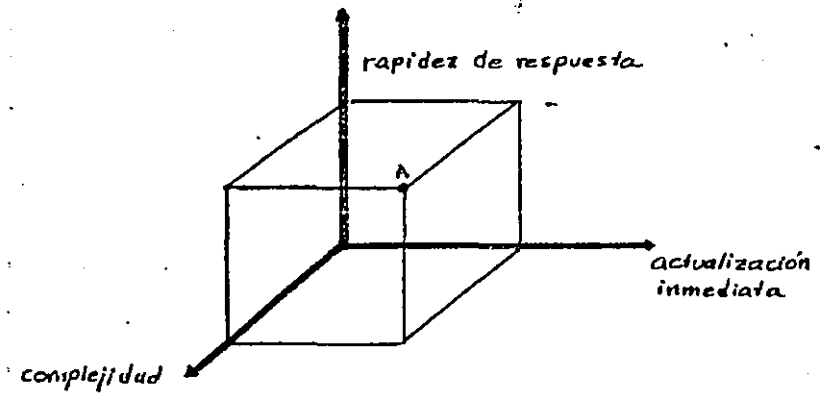
De esta manera, es posible, a través de un sistema general, obtener una respuesta aceptable, sin ser una implementación necesariamente óptima, pero si con una gran flexibilidad

en cuanto a la variedad de aplicaciones a implementar y con un grado mínimo de dependencia de los programas a una máquina en particular.

Características de un sistema general.

Las aplicaciones de las bases de datos se pueden dividir en dos grupos, sistemas de consulta y sistemas de proceso. Estas dos características se reflejarán básicamente en los factores: complejidad en las preguntas para consulta, actualización inmediata y rapidez de respuesta. El primer factor con el tercero nos da un sistema eficiente de información en línea, mientras que el segundo con el tercero nos da un sistema de procesamiento en línea y por último, el primero con el segundo nos dá un sistema donde el tiempo de respuesta no cuenta (como los sistemas batch).

Estos tres factores se pueden representar por tres ejes como se muestra en la figura siguiente:



SISTEMAS DE INFORMACION

FIG. II.A.

De esta forma un sistema general (considerando los factores rapidez de respuesta, complejidad en la pregunta y actualización inmediata).

Se situaría en el punto A de la figura.

La independencia de la máquina no es totalmente posible, ya que la implementación física de la base de datos se tendrá que hacer con las características físicas del tipo de máquina usado. Sin embargo, esta parte es pequeña en comparación a la parte lógica del sistema, la cual sí puede ser independiente de la máquina si el sistema es programado con un lenguaje de alto nivel y común a la mayoría de las máquinas.

Resumiendo, las características que deben cumplir un sistema general son las siguientes:

- Independencia de máquina, lo que implica:
  utilización de un lenguaje de alto nivel común a la mayoría de las máquinas con el cual se porgramará la parte lógica del sistema.
- Independencia de las aplicaciones, lo que implica:
  -capacidad de resolver preguntas complejas como un sistema de consultas.
  -capacidad de actualización inmediata como un sistema de procesamiento de datos.
  -rapidez de respuesta que permita operar al sistema además de Batch en línea.

II.s Sistema Reconfigurable.

Sobre sistemas generales ya se ha desarrollado bastante, siendo uno de los principales defectos de estos sistemas, la ineficiencia en la utilización de dos recursos (tiempo de respuesta y memoria requerida), en cierto tipo de aplicaciones muy específicas o por cambios en la operación del sistema; todo esto debido a la falta de correspondencia entre organización lógica y física de la información.

De este problema, ineficiencia-optimización de recursos, nace la idea de la reconfiguración como un medio de evaluar y adaptar la estructura física a la aplicación en particular del sistema, con el objeto de optimizar los recursos.

Uno de los problemas que plantea un sistema con tal característica, es que en un momento dado la solución (evaluación y reconfiguración) resulte más costosa que en un sistema con estructura invariable.

En el transcurso del trabajo se evaluará y se implementará este medio de optimización.

Primero se verá la estructura lógica que soportará el sistema, después la estructura física implementada, posteriormente la forma de operar la base, después un capítulo para la reconfiguración, y otro que describe como usar el sistema y por último un ejemplo implementado con resultados y observaciones.

III ESTRUCTURA LOGICA

III.1 Elementos básicos de información del sistema.

Este sistema contempla varias unidades de información (archivos, registros, propiedades, identificadores) que podrán ser accesados para su operación.

En la siguiente figura ilustraremos el conjunto de unidades de información que maneja este sistema:

ELEMENTOS DE INFORMACION
DEL SISTEMA
FIG. III. I. A.

En general, una base de datos se representará por un conjunto de archivos, donde cada archivo será un conjunto de registros, y a su vez estos, estarán formados por un conjunto de valores denominados propiedades.

Estos valores de propiedad para poder ser operados requieren de un método de obtención (método de acceso). A continuación se describen cada uno de los elementos de información que forman la base.

III. 1.1 PROPIEDADES

Las propiedades son el conjunto de características que forman un registro, y se definen como la unidad de datos mínima operable.

Por ejemplo, la propiedqd nombre del registro alumno es una característica, cuyo valor puede ser operado (la propiedad nombre del registro X es igual a Juan?).

Los valores en una propiedad, deben cumplir con la siguiente regla: la propiedad solo puede contener un solo valor, y se definirá por el tipo de valor (numérico o alfanumérico) y por la cantidad de valores posibles.

III.1.2 PROPIEDADES

Propiedad es el nombre de una de las cualidades de un registro, y se define como la unidad de datos mínima operable. Por ejemplo, la propiedad nombre del registro alumno es una característica de tal registro y su valor puede ser operado

(por ejemplo ¿el valor de la propiedad nombre del registro X is igual a Juan?).

El valor de una propiedad debe de cumplir la siguiente regla: la propiedad solo puede contener un valor, y se definirá por el tipo de vabr numérico y alfanumérico, y por la cantidad de valores posibles (clasificación que se verá más adelante).

III.1.1.1 IDENTIFICADORES

Un identificador es un tipo especial de propiedades cuya característica es que su valor para un registro determinado, lo hace diferente de los demás registros del archivo. De tal forma, es a través del identificador la manera de accesar un registro de un archivo.

El número de valores que puede contener esta propiedad, debe ser al menos igual al número de registros que componen al archivo.

Por ejemplo, es claro que la propiedad nombre puede definirse como identificador, puesto que hace único a cada registro, y el número posible de valores debe ser al menos igual al número de registros en el archivo; por otro lado la propiedad edad, definida por ejemplo entre 15 y 30 años el valor 20 no puede ser identificador, es muy probable que varias personas tengan 20 años en un archivo.

III.1.1.2 Tipos de propiedades en cuanto a sus valores.

Como se mencionó anteriormente las propiedades se definirán en cuanto al tipo y número de valor que pueden tener.

En cuanto al tipo de variables se definirán dos básicos, los numéricos (N) y los alfabéticos o alfanuméricos (A), por ejemplo: en cuanto al tipo de propiedad, edad es una propiedad numérica, a la cual se le pueden aplicar operaciones lógicas de comparación y aritméticas; nombre es una propiedad alfabética. Es importante hacer notar que las propiedades numéricas deben ser números enteros de punto fijo.

En cuanto a la cantidad de valores, pueden ser definidos (D), o indefinidos (I), por ejemplo la edad (en un archivo de alumnos) puede estar entre 18-30 (en total 13 valores posibles), y la propiedad nombre en cuanto a cantidad está indefinido. De esta manera agrupando estos dos grupos de características, darán un total de 5 tipos.

AI Alfabético indefinido

AD Alfabético definido

NI Numérico indefinido

ND Numérico definido

ID Identificador.

Por razones de implementación (será visto en el capítulo IV y V) se define un tipo más de propiedad, el tipo identificador (ID), que por su definición puede ser AI o NI.

De esta manera definidos los tipos de valores, las operaciones básicas que se pueden realizar son:

lógicas	$\in$	inclusión "está en"
	$\notin$	"no está en"
	$\sim$	negación lógica
	$\cdot$	"y" lógica
	$+$	"o" lógico
comparación	$P_1 = P_2$, $P_1 = cte$	
	$P_1 \neq P_2$, $P_1 \neq cte$	
	$P_1 > P_2$, $P_1 > cte$	
	$P_1 \geq P_2$, $P_1 \geq cte$	
	$P_1 < P_2$, $P_1 < cte$	
	$P_1 \leq P_2$, $P_1 \leq cte$	
aritméticas	$P_1 + P_2$, $P_1 + P_2$	
	$P_1 - P_2$, $P_1 - P_2$	
	$P_1 * P_2$, $P_1 * cte$	
	P_1 / P_2 , P_1 / cte .	

cte. constante

III.1.1.3 Propiedades compuestas y con multivalores

En la definición de propiedad se restringió, a que ésta sólo contenga un valor. Con dicha definición se eliminan en apariencia dos tipos de propiedades, que se pueden presentar en una base de datos, y estas son las compuestas y las propiedades con multivalores

III.1.1.3.1 Propiedades compuestas

Estas propiedades son las que tienen como "vabr" varios valores (también como unidades operables), por ejemplo, la propiedad fecha, está compuesta de 3 valores de propiedad más elementales, día, mes, año. En la siguiente figura se muestra el caso:

edad	fecha			----
23	a	m	d	----
	80	1	12	

propiedad compuesta (fecha)

edad	Fa	Fm	Fd	----
23	80	1	12	----

propiedad simple (fecha = fa año, fm fecha mes, fd día)

PROPIEDADES COMPUESTAS

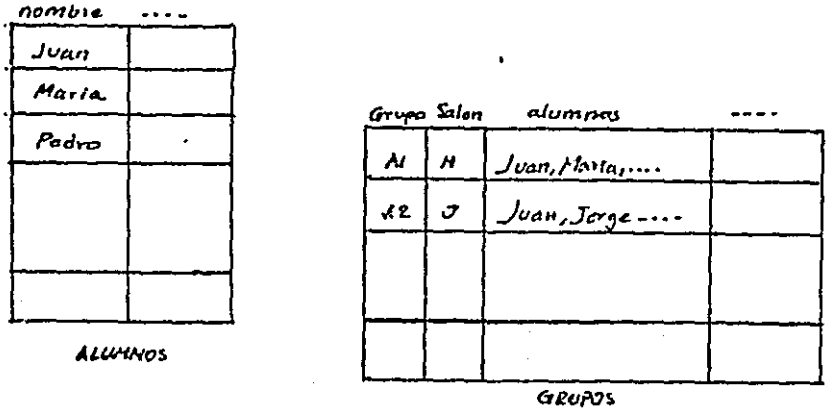
FIG.III.1.3.1.A

Cuando esta situación se presente, no se definirá fecha como una propiedad, se definirán tres propiedades FA, FM, FD (año, mes, día), de tal forma que nos de una unidad simple

y básica para ser operadas.

III.1.1.3.2 Propiedades con multivalores.

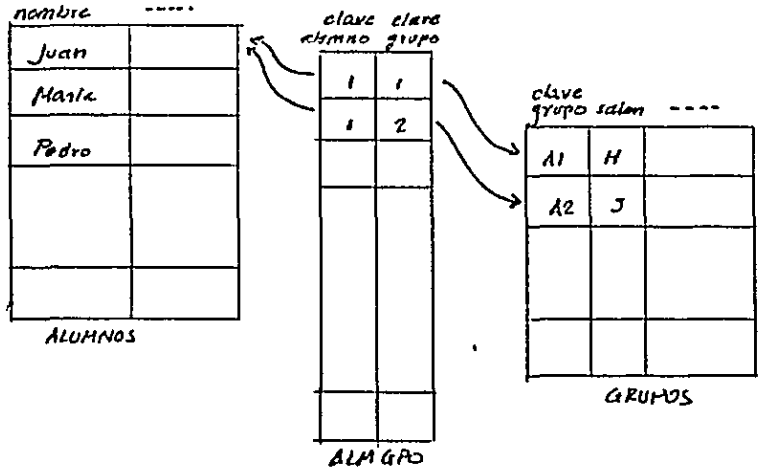
Este tipo de propiedades se presentan cuando para una propiedad dada se puede definir a un conjunto variable e indeterminado de valores para un registro en particular, por ejemplo, si tienen un archivo de alumnos y un archivo de grupos, dónde una de mis propiedades es, alumnos en el grupo. En la siguiente figura se ilustra el problema.



PROPIEDADES CON MULTIVALORES

FIG III. 1.1.3. 2. A

En esta propiedad, su valor no puede ser operado de manera directa (ya que está formado por un conjunto variable de valores), de tal manera que se requiere hacer alguna transformación para cumplir con valor único para propiedad. De esta manera la solución será:



ARCHIVOS SIN PROPIEDADES MULTIVALOR

FIG. III.1.1.3.2.13

Se crea un archivo intermedio ALMGPO, donde sus propiedades serán alumno y grupo (que no necesariamente tendrá que ser el nombre del alumno y el nombre del grupo, sino, va identificador, con el objeto de ahorrar memoria).

De esta forma la definición de propiedad se conservará y sus valores podrán siempre ser operados de manera directa.

Esta transformación es la usada para base de datos relacional. Referencia 1.

III.1.2 Registros

Los registros son las unidades de información de los archivos, donde cada registro está compuesto de un conjunto de propiedades.

Todos los registros de un archivo tienen el mismo número de propiedades, el cual siempre es fijo.

III.1.3 Archivos

Los archivos son las unidades de conformación de la base, donde cada archivo está compuesto de un conjunto de registros con igual número y tipo de propiedades. La formación de un archivo en cuanto a número de registros no tiene límite.

III.1.4 Relación entre archivos.

La base de datos total será descrita por la relación que exista entre los archivos. Esta relación estará dada a través de los identificadores. Por ejemplo si un archivo A tiene como propiedad el identificador de otro archivo B (nótese que la propiedad no es identificador de A y por lo tanto en su definición será propiedad no identificador), nos dará el medio para relacionar el archivo A y el B de tal forma que se podrán realizar operaciones sobre ambos archivos. Por

ejemplo, supongase un archivo de empleados de una empresa y un archivo de los diferentes departamentos de la empresa, en la siguiente figura se ilustrarán los archivos:

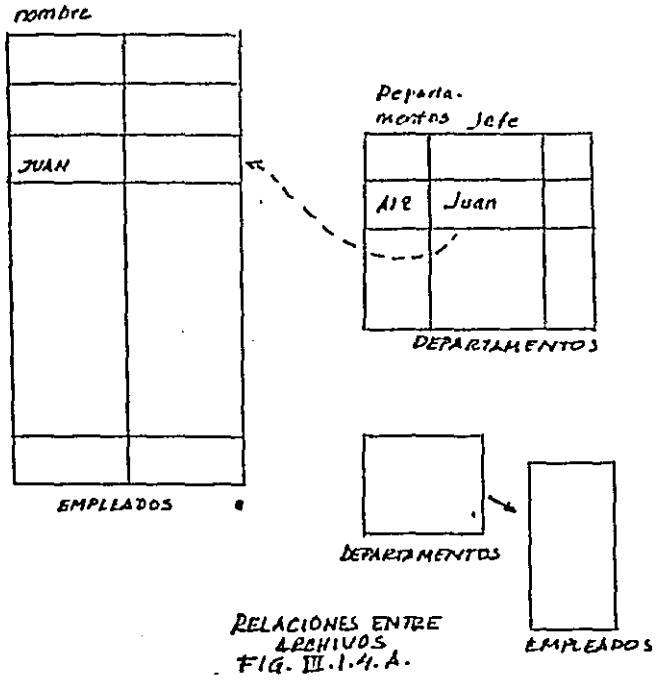


FIG. III.1.4.A.

El identificador del archivo empleados es el nombre.

El identificador del archivo departamentos es departamento, y una de sus propiedades jefe que compondrá a un nombre de empleado (y por lo tanto al identificador del archivo empleados).

De esta forma se puede hacer operaciones involucrando ambos archivos, por ejemplo, saber que jefes son menores de 30 años y tienen en su departamento a más de 20 empleados.

Esta operación se dividirá en dos partes, empleados menores de 30 años, viendo al archivo de empleados; y jefes (empleados) que tienen más de 20 empleados, viendo al archivo departamentos, haciendo un y lógico entre ambos subconjuntos tendremos el resultado.

En el capítulo de implementación se verá la forma de representar la estructura y las operaciones.

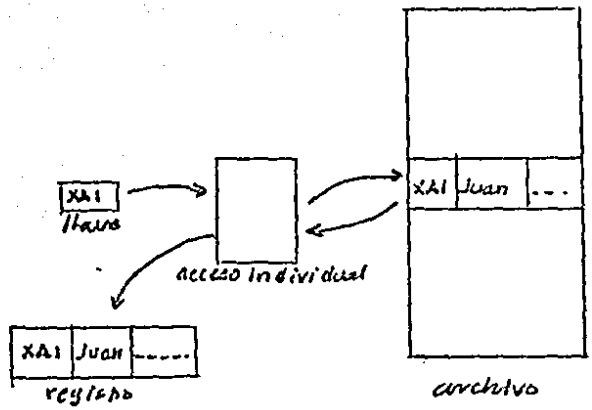
III.1.5 Método de acceso

El método de acceso es el mecanismo que permitirá obtener valores de propiedades para realizarles cierta operación.

Así, definido un conjunto de registros (archivo), habrá dos métodos de acceso; individualmente o por conjuntos. Es claro que la obtención, en cualquiera de las dos formas, está basada en los identificadores (características que hacen únicos a los registros de un archivo).

III.1.5.1 Acceso individual

Es el que permitirá obtener un registro en particular, del conjunto total, a través de uno de sus identificadores, de tal manera que se tendrá en un momento dado todas las característica (propiedades) de dicho registro, listas para ser operadas. En la sig. fig. se muestra este método de acceso

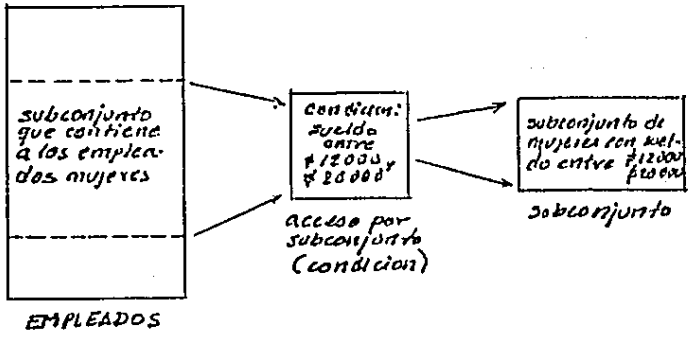


ACCESO INDIVIDUAL (POR LLAVE)
FIG. III.5.A.

III.1.5.2 Acceso por conjuntos,

Este tipo de acceso permite obtener del total de registros (o de un subconjunto de él), un subconjunto que cumple con una determinada condición (lógica, comparativa, aritmética) en los valores de algunas de sus propiedades. Es a través de esta operación que se pueden tener por ejemplo, los que tienen un sueldo de $12,000 y $20,000 de un archivo de empleados.

Para dar una idea más clara de este método de acceso usaremos la siguiente figura:



ACCESO POR CONJUNTO
FIG. III.5.B

En la figura anterior se muestra como se aplica el acceso por subconjunto; dado un subconjunto, (los empleados mujeres) se les pone una condición, (las que tengan el sueldo entre $12,000 y $20,000) y así se tiene un nuevo subconjunto, que está listo para ser operado o consultado.

Resumiendo, la estructura lógica que soporta el sistema implementado, es una organización de archivos del tipo modelo relacional (referencia 1), que brevemente podemos describir: todo archivo tiene que tener una representación matricial de dos dimensiones, (registros x propiedades). Donde los identificadores nos dán las ligas entre los archivos, y el sistema contará con dos métodos de acceso, individual y por conjunto.

Los registros podrán ser operados con funciones aritméticas, lógicas y de comparación sobre sus propiedades, con el objeto de poder ser consultados y/o modificados.

Es obvio que estas características definidas serán independientes de la estructura física en la cual se implemente el sistema.

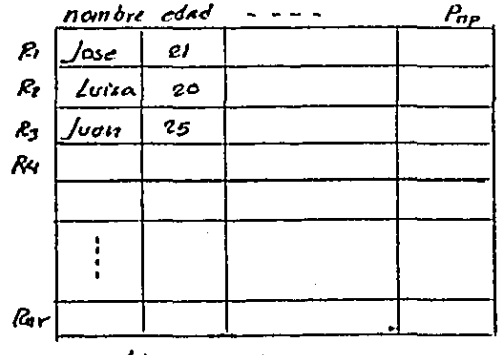
En el siguiente capítulo se describirá la implementación física de este modelo lógico.

IV. ESTRUCTURA FISICA

Una vez definida la organización lógica de la información, se definirá su estructura física, la cual deberá tener la flexibilidad suficiente para poder realizar cualquier tipo de operación.

A continuación se describe una estructura lógica y su correspondiente estructura física.

Existe una gran variedad de posibles estructuras físicas (archivos secuenciales, secuenciales por índice, aleatorios, invertidos, listas binarias, etc.), los cuales cada uno de ellos ofrecen ciertas ventajas y desventajas; para este trabajo, se eligieron los archivos secuenciales por índice y listas binarias, las cuales serán descritas en este capítulo.



AL- archivo
Pi- propiedades (np no. de propiedades)
Ri- registros (nr no. de registros)

ARCHIVO LOGICO AL

FIG. IV.A

IV.1 IDENTIFICADOR INTERNO

Cada registro lógico tendrá asociado un identificador único, para el manejo interno del sistema, el cual se denominará, identificador interno y es totalmente independiente del identificacor lógico.

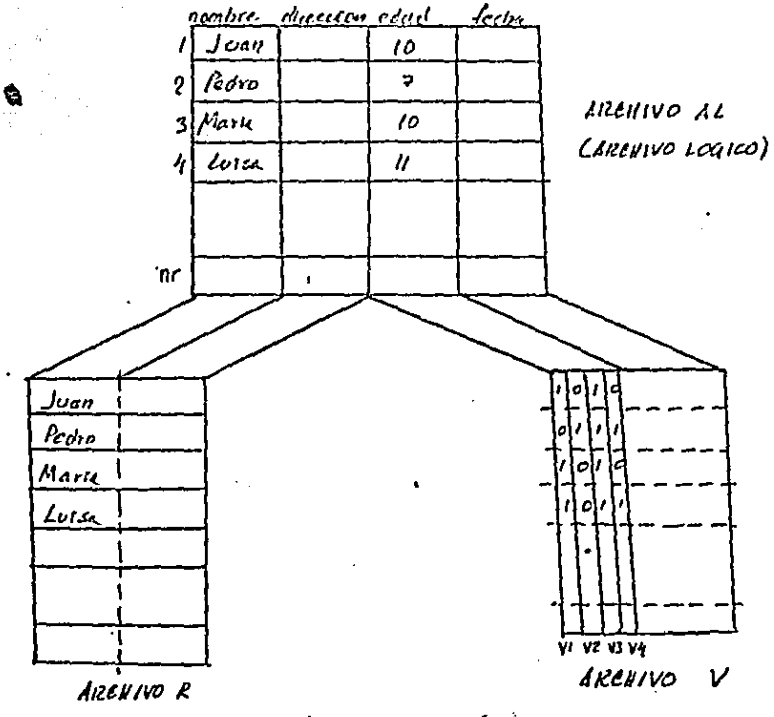
Este identificador se formará de manera secuencial y en el orden de aparición de cada nuevo registro en AL. Es decir, el primer registro tendrá como identificador interno al 1, el segundo al 2, etc., de esta manera los identificadores internos son implícitos, no ocupan lugar en memoria, siguen un orden secuencial y son independientes de cualquier orden de los registros de AL.

Por ejemplo, observando la figura anterior, el registro lógico Juan tendrá como identicador interno igual a 3.

La implementación de estos identificadores y su relación con el método de acceso por registro será visto en el inciso IV.4 Tablas Indices.

IV.2 Representación de propiedades.

En general todo archivo lógico se almacenará en dos archivos físicos R y V, para ilustrar la implementación de un archivo lógico se usará la siguiente figura:



$AL \Rightarrow R \text{ y } V$

IMPLEMENTACION FISICA DE AL (ARCHIVO LOGICO)

FIG. IV. 2A

Cada registro físico del archivo R contendrá un conjunto de propiedades correspondientes a cada registro del archivo AL. Por ejemplo, observando la figura R está formado sólo por dos propiedades de AL (nombre y dirección), así el registro 1 de R contiene el nombre y dirección del primer registro de AL, el registro 2 de R contiene la dirección y el nombre del segundo de AL y así sucesivamente.

Por otro lado, cada bit del registro físico del archivo V corresponde a un solo bit del valor de una propiedad para todos los registros del archivo. Por ejemplo, volviendo a la figura, en el archivo lógico se difinió para la propiedad edad valores entre 0 y 15 (se requiere de 4 bits para su representación binaria), esta propieddd requiere de 4 vectores binarios (registros físicos) de un archivo. De esta manera el primer registro (VI) del archivo V contiene el bit más significante de la propiedad edad para todos los registros.

IV.2.1. Archivos R (ventajas y desventajas)

Dado un archivo lógico AL y su correspondiente archivo R, las ventajas que este ofrece son las siguientes:

Bastará accesar un registro de este archivo para obtener un conjunto de valores de propiedades de una sola vez. Lo cual resulta ser muy útil, cuando exáctamente se tiene al registro X, y se desea operar sobre esas propiedades,

de tal forma que el procedimiento a seguir será:

- acceso al registro para ser leído
- operación sobre sus propiedades
- acceso para actualización (si se requiere)

Por ejemplo, supóngase que tenemos un archivo AL como el de la figura IV.2.A, y que todas las propiedades están en R, y se desea actualizar los valores de las propiedades dirección y fecha de ingreso de María (también se supone que existe un medio para accesar directamente a María). Para resolver esta operación bastará accesar al registro María, con lo cual tenemos de una sola vez el conjunto de propiedades, que se requieren, así con los valores obtenidos se operarán y finalmente se hará un acceso para actualizar la información, así podemos decir se requieren de dos accesos para una actualización en R.

Al igual que la operación de modificación antes mencionada, el procedimiento será semejante para las altas, bajas, y consultas a registros a través de su identificador.

Sin embargo, si la operación no es sobre el valor de una propiedad de un registro en particular, y es sobre todos los valores de una propiedad (o varias) de un archivo, la estructura R no resulta ser la más adecuada, puesto que se tendrá que leer todo el archivo R, registro por registro para aplicar la condición al valor de propiedad requerida y ver si lo cumple o no. Esta operación resultará ineficiente en

cuanto a cantidad de información accesada, pues de todos los valores accesados, en cada registro R, sólo se están realmente ocupando un valor (o varios), Por ejemplo, volviendo al archivo anterior, si la operación consiste en conocer quienes tienen edad 10, la respuesta, con estructura R, sólo podrá ser dada leyendo todo el archivo R, donde a cada registro accesado, se consultará el valor edad para ver si es igual a 10, sin haber utilizado los demás valores de propiedad, lo cual no resulta óptimo en información accesada y por lo tanto, en tiempo de respuesta, puesto que si para conocer una sola propiedad se requiere el acceso de toda la información el tiempo de respuesta se puede ver muy afectado, por ejemplo si solo se requiere la edad de un archivo de 10 propiedades, aproximadamente el 90% de la información no se ocupó, pero sí tomó tiempo el obtenerla.

De esta manera una consulta sobre todo el archivo requiere de NR consultas al archivo R, donde NR=número de registros en el archivo.

IV.2.2. Arvhivos V (ventajas y desventajas)

Una propiedad representada en V, tendrá tantos registros (vectores) en este archivo como el número de Bits máximo que se requieren para expresar el valor de la propiedad en Bits.

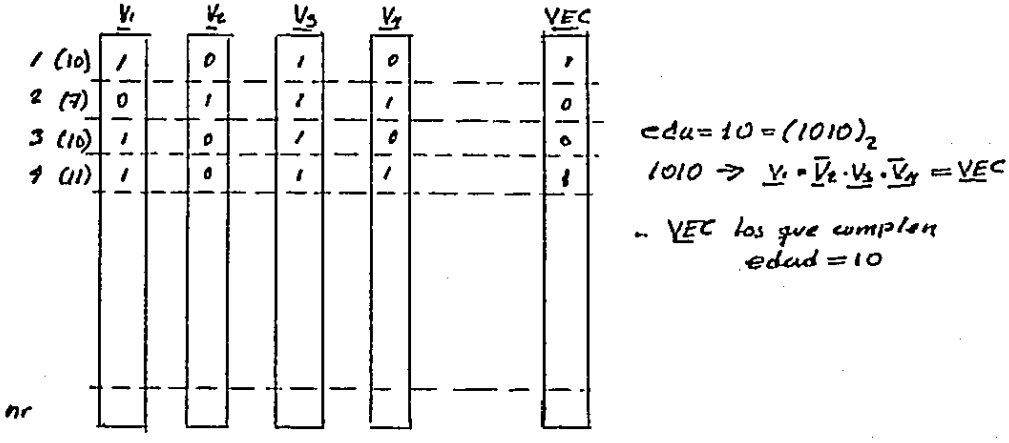
De esta manera, en operaciones que incluyan todo el archivo, basta con hacer ciertas operaciones sobre el conjunto

de vectores binarios involucrados sin necesidad de accesar otra información no necesaria (como en el caso de R).

Por ejemplo volviendo a la pregunta anterior (edad-10) pero ahora suponiendo que esta propiedad está en V, como se mostró en la figura IV.2.A estará representada por los vectores 1, 2, 3 y 4. Por lo tanto para la condición edad=10 bastará hacer la siguiente operación lógica bit a bit de cada vector como sigue:

$$\underline{V_1} \cdot \overline{\underline{V_2}} \cdot \underline{V_3} \cdot \overline{\underline{V_4}} \quad (1010) \qquad \underline{V_i} - \text{vector}$$

Con lo cual se obtendrá como resultado un vector binario, donde por medio de sus "1"s indicará quienes cumplen con edad=10. El vector resultado (vector de acceso) de la operación, se muestra en la siguiente figura.



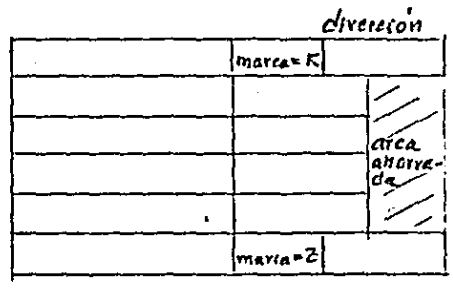
VECTOR DE ACCESO

FIG. IV.22.A

Propiedad explícita.

Volviendo a la figura anterior se observa que el valor marca solo lo contienen dos registros del total de 6 y se observa que habría un desperdicio de memoria si se usara esta implementación.

Otra posibilidad de representación es:



PROPIEDAD EXPLICITA
FIG. IV.2.C

Es decir, una propiedad explícita origina que se almacenen los datos en el registro: el nombre o identificador de la propiedad (la palabra "MARCA") y el valor de esa propiedad (el entero K). La posición donde esté guardado este par de

Por lo tanto de manera general se puede decir que en una operación de consulta se requiere accesar sólo el número de vectores que participan en la operación, así si una propiedad puede tener 20 valores, se requieren 5 bits, mínimos para una representación binaria ($2^5 \geq 20$), es decir en general se requieren de $\log_2$ VALMAX accesos, donde VALMAX = al no. maximo de valores en la propiedad.

De manera inversa que R, el problema de las propiedades en archivo V se presentará, cuando la operación es sobre unas propiedades de un elemento en particular. Mientras que en una estructura R bastará un solo acceso para tener todos los valores asociados al registro, en un archivo V para accesar un solo valor de propiedad, se requerirán de accesar todos los vectores que la componen. En este caso el número de accesos será:

$\log_2$ (VALMAX) VLAMAX-valor máximo que puede tener la propiedad.

y si se hace una actualización se le agregará un factor de 2.

Resumiendo las características de los archivos R y V, se observa que la forma de operar es contraria en cuanto a accesos, de tal forma que para un cierto tipo de operación una estructura es óptima y para la otra estructura no lo es. En la siguiente gráfica se muestran las características

OPERACION / ESTRUCTURA	ACTUALIZACION *	CONSULTA *
R	2	NR
V	2 * MAXVAL	MAXVAL
RV	2	MAXVAL

NR = NO. DE REGISTROS EN EL ARCHIVO

MAXVAL = VALOR MAXIMO QUE PUEDE TENER LA PROPIEDAD

* UNIDAD = "ACCESO", (ACCESO REGISTRO = ACCESO VECTOR)

RV - PROPIEDAD EN AMBAS ESTRUCTURAS

COMPARACION DE LAS ESTRUCTURAS

FIG. IV.2.2.B

En la tabla anterior se comparan la estructura R, V y la combinación de ambas en cuanto al número de "accesos" rqueridos. Dicha unidad, "accesos", se hace considerando que toma el mismo tiempo en accesar un registro físico de R (renglón) y un registro físico de V (vector), lo cual es aproximadamente cierto, pero útil como medida de comparación (por ejemplo, un archivo de 10,000 registros y 200 palabras/-48 bits por registro "accesos" en R y V son equivalentes). Se puede observar también, que una estructura RV tiene la ventaja, en cuanto "accesos" de tener las mejores cualidades de R y V, pero su precio está en la memoria, utiliza el doble, puesto que la información está repetida en R y V.

IV.3 Implementación de propiedades.

Una vez definidas las ventajas y desventajas de los archivos R y V, es claro que las propiedades que deberán estar en R serán aquellas, donde las operaciones más importantes que se hagan sean sobre registros renglones, en particular como son las actualizaciones. Las propiedades que deberán estar en V serán aquellas cuyas principales operaciones sean consultas incluyendo a todos los registros en un archivo (consulta global), consulta de propiedades. De esta manera, puede decirse de forma general que:

actualizaciones - - - → archivo R

consulta global - - - → archivo V

Sin embargo, dado que el sistema planteado es general, no se sabe que tipo de operación (actualización o consulta global) predominará durante su uso.

De alguna manera en el momento de creación de la base se puede saber el tipo de aplicación (tipo de operaciones más frecuentes) a la cual estarán orientadas las propiedades.

Esta información puede ser obtenida de dos formas:

implícitamente (intuitivamente)

explícitamente

IV.3.1 Implementación implícita.

En este caso se parte de que no se conoce el comportamiento de las propiedades en cuanto a operaciones; sin embargo, por las características particulares de las propiedades se puede intuir el tipo de aplicación a la cual estará orientada.

En el capítulo de estructura lógica se definieron los tipos diferentes de propiedades en cuanto a los valores que pueden tener.

1.- AI. Valores alfanuméricos indefinidos

2.- AD. Valores alfanuméricos definidos

3.- NI. Valores numéricos indefinidos

4.- ND. Valores numéricos definidos.

En los tipos 1 y 3 al haber un gran número de valores

posibles, se deduce que es muy factible que existan valores casi únicos para cada propiedad, de tal forma que generalmente las operaciones serán sobre un registro en particular.

Por ejemplo la propiedad dirección en un archivo de empleados será generalmente un valor único para cada registro empleado (podría ser igual, en el caso que dos o más empleados vivieran en la misma casa o condominio). Por lo que es más factible que las operaciones más comunes sean del tipo: obtener la dirección de un empleado en particular.

En el caso de los tipos de propiedad 2 y 4 la situación es diferente, si el número definido de valores es menor o mucho menor que el número de registros en el archivo, muchos de ellos contendrán valores iguales. Por ejemplo, la propiedad edad en el archivo de empleados (considerando que debe estar entre 18 y 60) y suponiendo que en total sean unos 2,000, habrá varios que tengan edad=25. De esta manera se puede pensar que sobre este tipo de propiedades las operaciones estarán orientadas al tipo siguiente: "quienes cumplen con edad 20", y no sobre un registro definido, ya que este valor no identifica a alguien en particular, puesto que un valor lo contienen varios registros.

Bajo estas suposiciones las propiedades en cuanto a su tipo, se estructurarán de la manera siguiente:

AI $\longrightarrow$ R

AD $\longrightarrow$ V

NI $\longrightarrow$ R

ND $\longrightarrow$ V

Si las suposiciones antes citadas son ciertas, las operaciones sobre las propiedades serán óptimas, puesto que estarán orientadas a su aplicación; sin embargo, en la realidad estas suposiciones pueden no cumplirse y tener por lo tanto una "ventaja" realmente como una desventaja.

De esta manera, aún con las mejores intenciones, la estructura puede fallar. Así la solución tendrá que ser dinámica y adaptarse al tipo de aplicación que un momento dado tenga el sistema. En el capítulo de reconfiguración se verá más ampliamente la adaptación dinámica del sistema.

IV.3.2 Implementación explícita

En este caso se supone que el que genera la base conoce cual es el comportamiento de las propiedades, de tal forma que lo informará al sistema, en el momento de creación, que tipo de aplicación tendrá cada propiedad (y el sistema de acuerdo a esto estructurará la propiedad).

Sin embargo, esta estrategia puede fallar debido a que la aplicación puede cambiar en el futuro o simplemente por error humano en el momento de creación. Pero si el sistema se plantea reconfigurable cualquier error se corregirá en

el futuro (siempre y cuando el método de reconfiguración sea correcto).

Otra alternativa, es la combinación de ambas implementaciones, explícitas e implícitas; es decir, donde la información del usuario (si es que existe), se puede tomar en cuenta en la implementación.

IV.3.3. Propiedades implícitas y explícitas.

Otros tipos de implementación de propiedades que se puede plantear (no implementadas en este sistema por falta de tiempo) son las explícitas y las implícitas.

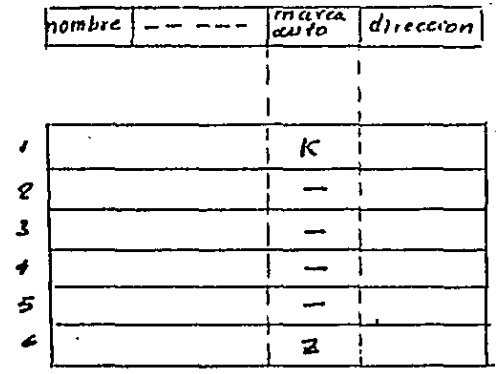
Estos dos tipos de propiedades se plantean ante el problema que se puede presentar sobre ciertas propiedades y que en determinado momento puede significar desperdicio de memoria.

Este problema se presenta cuando, los valores de una propiedad definida para un archivo no la contienen la mayoría de sus registros. Por ejemplo, en la propiedad marca de automóvil del empleado para un archivo de empleados, tal vez una gran mayoría no cuente con automóvil y por lo tanto en la mayoría de los registros esta información no exista.

Bajo esta situación esta propiedad puede ser implementada de dos formas, implícita y explícita.

Propiedad implícita

En este caso se define el valor de esta propiedad por la posición que ocupa de acuerdo a un formato de registro



PROPIEDAD IMPLICITA

FIG IV. 2.B

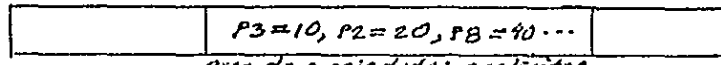
Observando la figura anterior, el valor de la propiedad marca aparece en R, pero no el nombre (marca) de la propiedad ya que no es necesario, puesto que el valor se identifica por la posición que ocupa.

ya no indica nada.

Sin embargo, el precio de este ahorro de memoria está en que la operación se hará un poco más complicada, por ejemplo se tiene que tomar en cuenta que si se dan de alta los valores no existentes, habrá que tener areas en memoria para estos valores.

Con este tipo de representación se pueden pensar en dos formas de organizar a un conjunto de propiedades que son explícitas (pensando en una area fija un poco mayor para posibles altas).

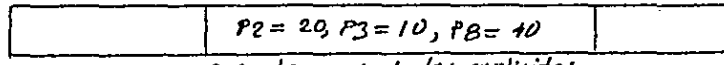
Supongase 3 propiedades explícitas que son representadas de la siguiente manera:



area de propiedades explicitas

Pi – propiedades

La otra posibilidad para estas mismas propiedades es:



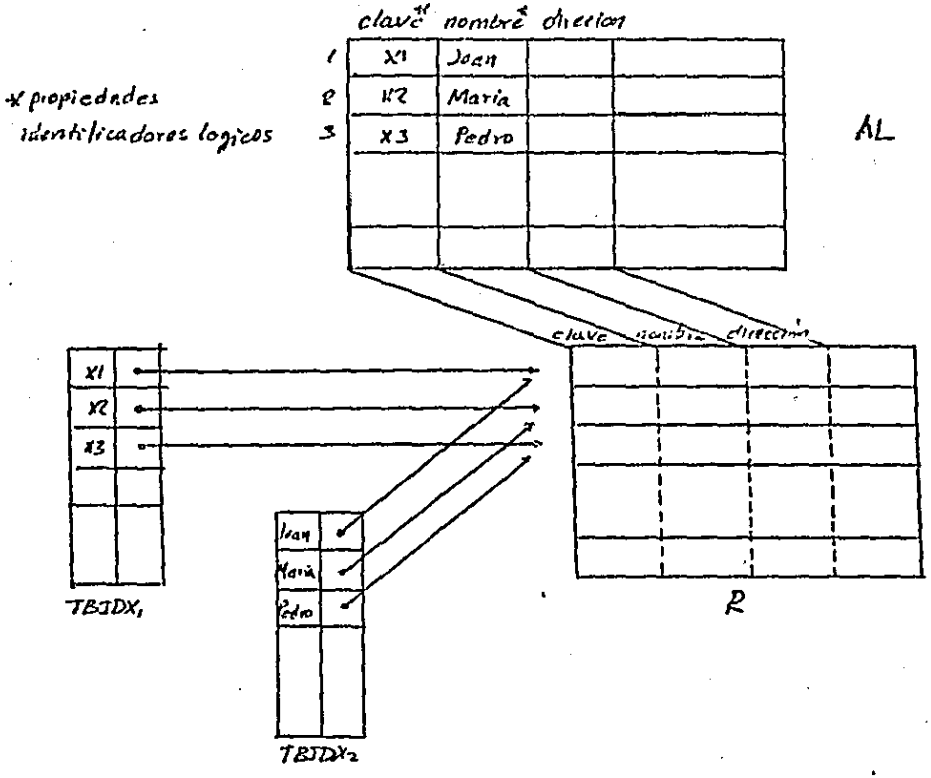
area de propiedades explicitas

Es decir se pueden tener las propiedades explícitas desordenadas u ordenadas con el único propósito de facilitar las operaciones. Si están ordenadas, la consulta se hace con una búsqueda binaria. Si están desordenadas, para saber si un registro contiene o no una propiedad hay que efectuar una búsqueda secuencial sobre el área de propiedades explícitas.

IV.4 Tablas Indices (acceso directo).

En el capítulo de estructura lógica, además de los cuatro tipos de propiedades antes referidos (AI,AD,NI, y ND), se mencionó la propiedad tipo identificador (llave de acceso lógico), la cual por su naturaleza puede estar incluida dentro de los tipos 1 o 3, por que existía no un número indefinido de valores pero sí un valor único para cada registro. Pero se da como un tipo diferente porque tendrá una función especial. Primero servirán de ligas entre los archivos de tal forma que permitirán hacer operaciones entre archivos (esto será más ampliado en el capítulo de operación de la base). Segundo, este tipo de propiedad permitirá el acceso directo a través de las tablas índices que se crearán a partir de su identificación.

Toda propiedad definida como identificador lógico tendrá una correspondencia uno a uno con el identificador interno del archivo. Esta correspondencia será lograda a través de la tabla índice, lo cual se presenta con la figura siguiente:



TBIDX1 Tabla indice para acceso por clave

TBIDX2 Tabla indice para acceso por nombre

TABLAS INDICE

FIG. IV.4.1.

De la gráfica anterior se observa que toda propiedad identificadora además de estar representada en un archivo R creará una tabla índice que permitirá a través del identificador lógico conocer su correspondiente identificador interno y por lo tanto su posición física dentro del archivo R.

Esta estructura tiene las ventajas siguientes:

Independencia del identificador lógico con el interno, lo cual, facilita y simplifica las operaciones.

Da un medio de acceso directo a registros, que facilita las operaciones tanto actualizaciones como consultas globales.

La única restricción del sistema en cuanto a los identificadores, es que no se puede modificar el contenido de una llave, por evitar una incompatibilidad de información.

De esta manera el acceso de un registro consiste en un acceso (lectura a memoria principal de toda la tabla índice) y de una búsqueda (de la llave deseada en la tabla).

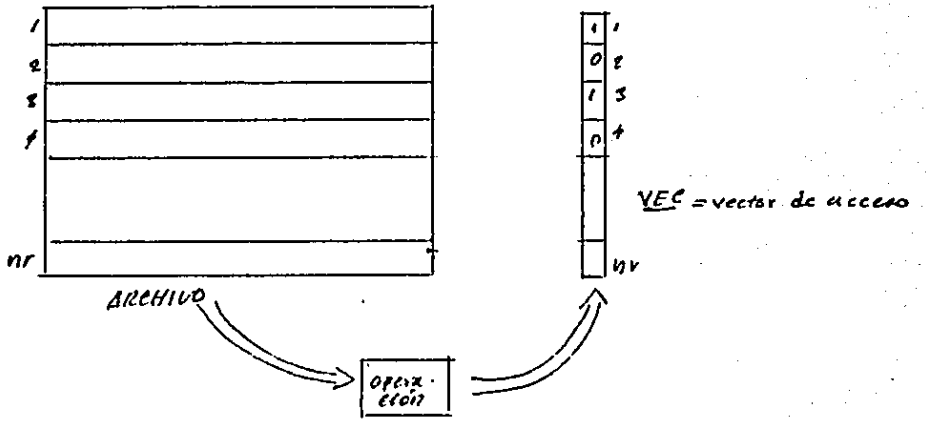
Cuando la memoria principal es reducida la lectura de la tabla índice constará de dos accesos, una subtabla (indicando en que parte de la tabla índice se podrá encontrar la llave) y el acceso de esa parte de tabla índice.

IV.5 Vectores de acceso.

Como se vió en el inciso IV.2 a través de un ejemplo, el resultado de una operación global (que será visto más

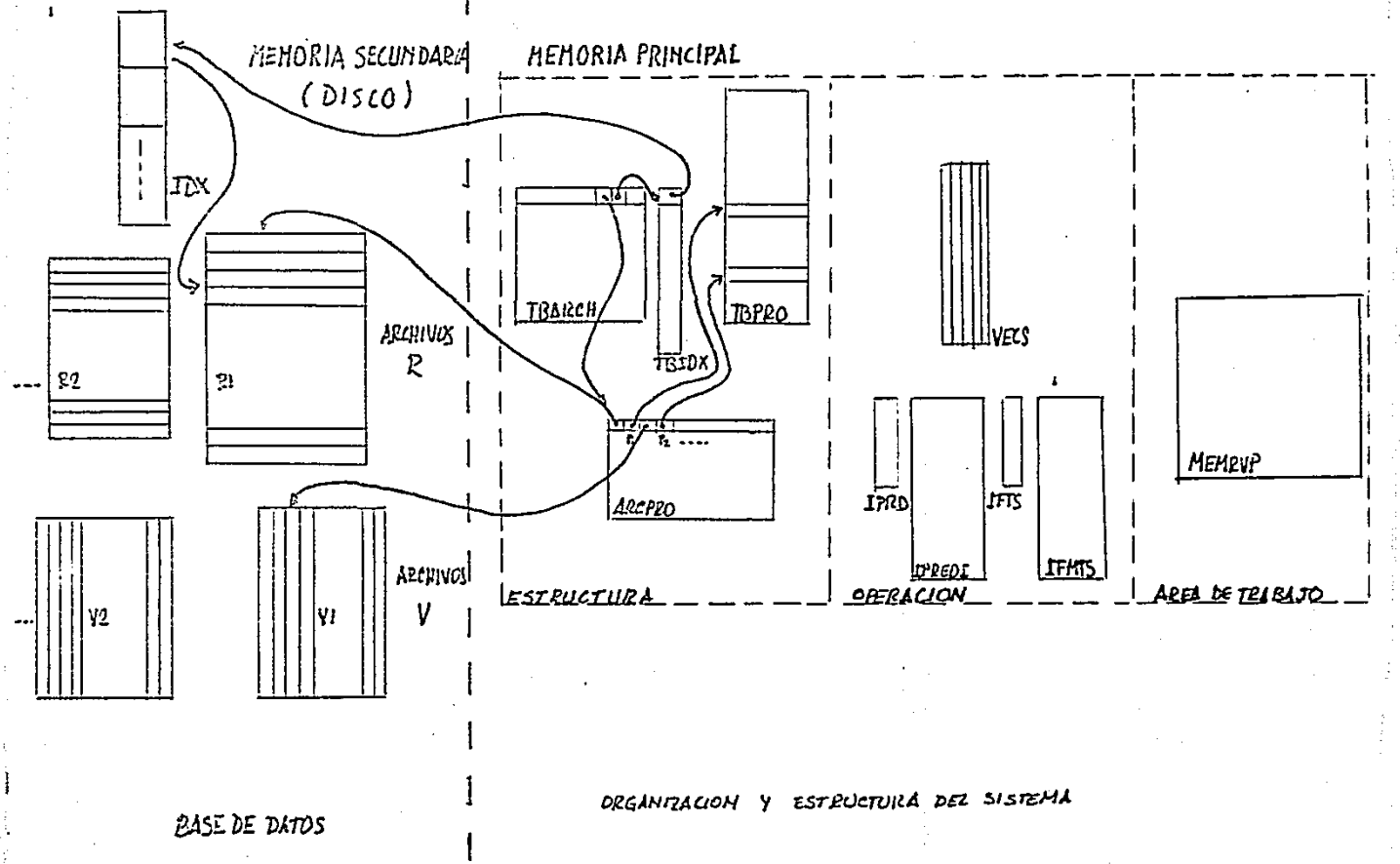
ampliamente en el capítulo V), es un vector de acceso, el cual consiste de un registro con un número de bits igual al número de registros del archivo al cual accesa, y cuya característica principal es que los bits "1"s dentro del registro dan a través de su posición, el número de registro (= a su identificador lógico) a ser accesado. Por esta razón se utiliza el identificador interno secuencial, por el ahorro de memoria que se obtiene por su utilización, puesto que implícitamente nos da la dirección a través de la posición de los bits "1".

Así un vector de acceso puede ser expresado como:



VECTOR DE ACCESO

FIG. IV. 5.2



ORGANIZACION Y ESTRUCTURA DEL SISTEMA

FIG. III.6.A

IV.6 Estructura del sistema implementado

La base de datos del sistema se divide en dos partes, la base de datos en sí y que reside en memoria secundaria, con acceso directo (como disco, tambor, etc.), y la que contiene la estructura y organización de la base que reside en memoria principal. Esta última a su vez, se divide en tres partes: estructura, operación y áreas de trabajo.

ESTRUCTURA

La sección de estructura, es donde reside toda la información acerca de la organización de los datos, aquí se encuentran todas las direcciones físicas de los datos de la base (archivos, R.V., etc.). Básicamente esta sección está compuesta por tres tablas principales, la tabla de archivos lógicos de la base (TBARCH), la tabla de propiedades en la base (TBPRO) y la tabla que relaciona archivos y propiedades de la base (ARCPRO). A continuación se describen cada una de las tablas de esta sección.

TBARCH.

Esta tabla contiene toda la información referente a todos los archivos lógicos existentes en la base, como son: identificador lógico, identificadores internos de sus archivos, R y V correspondientes, número de registros lógicos, etc.

A continuación se describen cada uno de los elementos que forma esta tabla.

TBARCH

Tbarch 1	IDLAR (identificador lógico de archivo)
Tbarch 2	IR (Dirección del archivo R)
Tbarch 3	IV (Dirección del archivo V)
Tbarch 4	NR (No. de registros en el archivo lógico)
Tbarch 5	NV (No. total de vectores en el archivo V)
Tbarch 6	INI (Apuntador a la tabla ARCPRO)
Tbarch 7	IFI (apuntador a la tabla ARCPRO)
Tbarch 8	Sin uso
Tbarch 9	Sin uso
Tbarch 10	Sin uso
Tbarch 11	NP (No. de propiedades en el archivo)
Tbarch 12	AI (Apuntador a la tabla indice)
Tbarch 13	NI (No. de identificadores en el archivo)
Tbarch 14	NBR (No. de bits en R)
Tbarch 15	NBV (no. de bits en V)

Esta tabla en particular apunta a una subtabla TBIDX la cual contiene los identificadores lógicos que tiene cada archivo. El objeto de esta tabla es conocer la dirección (a través del identificador lógico) de la tabla indice correspondiente al archivo I que está residente en memoria.

A continuación se describe la tabla:

TBIDX

Tbidx 1 NIL (Nombre del identificador lógico)

Tbidx 2 I1 (Apuntador al archivo de tablas)

Tbidx 3 I2 (Apuntador al archivo de tablas)

Tbidx 4 No usado

Esta tabla contiene todas las propiedades existentes en la base así como sus características principales como: nombre lógico, identificador interno, tipo de propiedad, etc. Esta tabla está formada por los siguientes elementos:

TBPRO

Tbpro 1 ILP (Identificador lógico de propiedad)

Tbpro 2 No usado

Tbpro 3 TPV (Tipo de valor)

Tbpro 4 U (Unidad del valor)

Tbpro 5 LINF (Límite inferior del valor)

Tbpro 6 LSUP (Límite superior del valor)

Tbpro 7 INC (Incremento en el valor)

Tbpro 8 FUNC (Identificador de función para checar)

Tbpro 9 - Tbpro 15. no usados

ARCPRO

Esta tabla contiene las relaciones que existen entre el conjunto de archivos y el conjunto de propiedades. En otras palabras, contendrá básicamente las imágenes, en cuanto

a formato, de todos los archivos lógicos existentes en el sistema. De tal forma que para llevar a cabo una operación de una determinada propiedad de un determinado archivo, se hará uso de esta tabla, pues aquí se encontrará la localización de cualquier propiedad dentro de los archivos R y/o V. Además esta tabla contendrá la información estadística del comportamiento de las propiedades. La información que contiene esta tabla es la siguiente:

	ARCPRO
Arcpro 1	IA (Identificador interno de archivo)
Arcpro 2	IP (Tipo de propiedad)
Arcpro 3	F1 (Especificación de formato)
Arcpro 4	F2 (Especificación de formato)
Arcpro 5	ESTR (Estructura R, V o RV)
Arcpro 6	EXI (Propiedad explícita o implícita)
Arcpro 7	DIRV (Si en V dirección)
Arcpro 8	NBP (No. de Bits en la propiedad)
Arcpro 9	FOPA (Frecuencia de operación en actualizaciones)
Arcpro 10	FOPC (Frecuencia de operación en consultas)
Arcpro 11	FOPC' (Frec. de Op. en R estando en V)
Arcpro 12	PS (Posición de la propiedad en R)
Arcpro 13	ARCPRO-15 no usados

OPERACION

En esta sección de memoria estarán los arreglos que se requieren para definir las operaciones que se harán sobre la base. Esta área básicamente tendrá las tablas para definir los predicados (IPRD, IPREDI), las salidas (IFTS, IFMTS) y un conjunto de vectores para operaciones intermedias entre vectores (Vecs).

A continuación se describen cada una de las tablas:

IPRD, IPREDI

En la tabla IPRD estarán contenidos todos los predicados que se pueden generar en un momento para efectuar operaciones de consulta global. En la tabla IPREDI estarán las expresiones que forma cada predicado. A continuación se describen los elementos de ambas tablas:

	IPRD
Iprd 1	IDP (Identificador de predicado)
Iprd 2	IDAR (Identificador interno del archivo)
Iprd 3	INIP (Apuntador a la tabla IPREDI)
Iprd 4	IR (Apuntador a la tabla IPREDI)
	IPREDI
Ipredi 1	OPDOR (Operador)
Ipredi 2	OPDO1 (Operando 1)
Ipredi 3	OPDO2 (Operando 2)
Ipredi 4	OPDO3 (Operando 3)
Ipredi 5	TPV (Tipo de variables)

IFTS, IFMTS.

Esta es una estructura semejante a la anterior, en la tabla IFTS están contenidos todas las funciones de salida que en un momento se puedan operar y en la tabla IFMTS estará la función descrita en detalle. Los elementos que la componen son:

IFTS

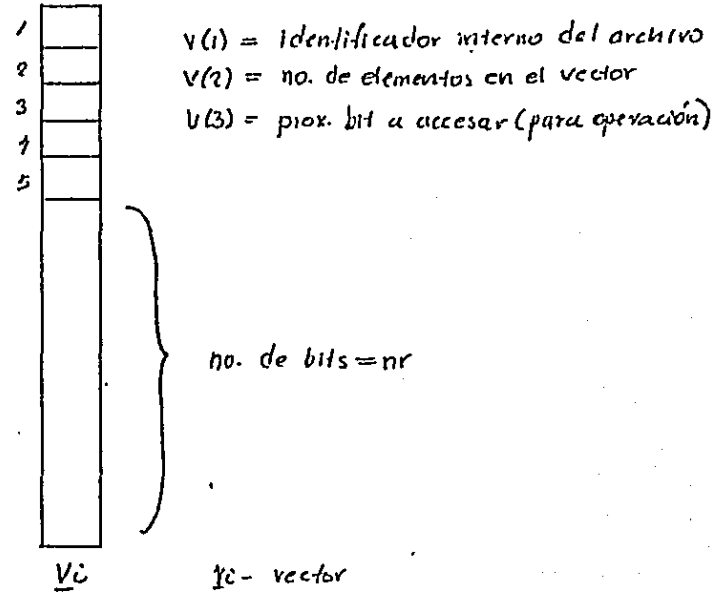
Ifts 1	IDS (Identificador de salida)
Ifts 2	IAR (Identificador de archivo)
Ifts 3	INIS (Apuntador a la tabla IFMTS)
Ifts 4	IS (Apuntador a la tabla IFMTS)
Ifts 5	AS (Archivo de salida)

IFMTS

Ifmts 1	IDP (Identificador de propiedad)
Ifmts 2	NC (No. de campos)
Ifmts 3	TC (Tipo de campos)
Ifmts 4	ESP (espacios)
Ifmts 5	ENRV (Tipo de estructura R o V)
Ifmts 6	POS (Posición dentro del registro)
Ifmts 7	NBSP (No. de bits en la propiedad)

VECS.

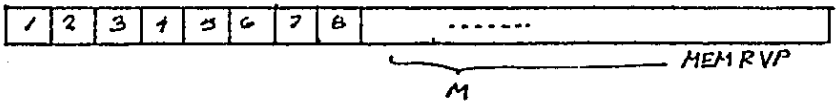
Esta tabla es simplemente un conjunto de vectores que estarán listos para realizar cualquier operación intermedia entre vectores. El formato de cada vector es mostrado en la siguiente figura:



FORMATO DE UN VECTOR
FIG. IV. 6. B

Además del area de operación se requiere de una area de trabajo necesaria para cierto tipo de operaciones que involucren las estructuras R y V. El manejo de esta area será ampliado en el capítulo de operaciones sobre la base, pero la idea básica es tener una matriz de dimensión dinámica que permita pasar parte de archivo R (Residente en disco) a una matriz en memoria (MEMRVP) con el objeto de transformarlo a un archivo V y viceversa.

A continuación se describe el formato de esta matriz dinámica:



MEMRVP 1 = identificador del area
2 = no. de renglones en la matriz
3 = no de vectores de la matriz
4 = tamaño del renglon
5 = tamaño del vector
6 = proximo renglon
7 = proximo vector
8 = no usado
M = area disponible para formar la matriz dinamica

MATRIZ DINAMICA MEMRVP
FIG. IV. 6.C.

Dirección física de este archivo también está dado por TBARCH y los vectores correspondientes a una propiedad en particular está dado por ARCPRO.

TABLAS INDICES

Este archivo contendrá todas las tablas indices de los archivos de la base. Por medio de esta tabla será posible accesar a un registro a través de su identificador lógico.

V OPERACIONES SOBRE EL SISTEMA

V.1. Métodos de acceso

Antes de poder realizar cualquier operación sobre los datos de la base, es necesario resolver un problema más fundamental, el acceso a los datos, que se denominará método de acceso.

Para este sistema se difinirán dos métodos de acceso:

acceso directo a registro

acceso por propiedad o acceso vectoral

(a un conjunto de registros)

V.1.1. Acceso directo a registro.

Este método de acceso permitirá hacer operaciones sobre las propiedades de un registro en particular de un archivo, para lo cual se hará uso de la: característica. que hace único a cada registro, su identificador.

De esta forma el sistema cuenta con una función básica de acceso a registro:

DAREG (ARCH,PIDENTI=VAL, REG , E)

ARCH- Nombre del archivo a accesar

PIDENTI- Propiedad identificador

VAL- Un valor de esa propiedad, que identifica a un registro en particular.

REG- Arreglo donde se obtendrá toda la información del registro

E - Código de error si no se pudo realizar la operación

Una variante de esta función, es donde solo se accesa el valor de una propiedad.

VALP=PAPROP (ARCH,PIDENTI,VAL,PROP, E)

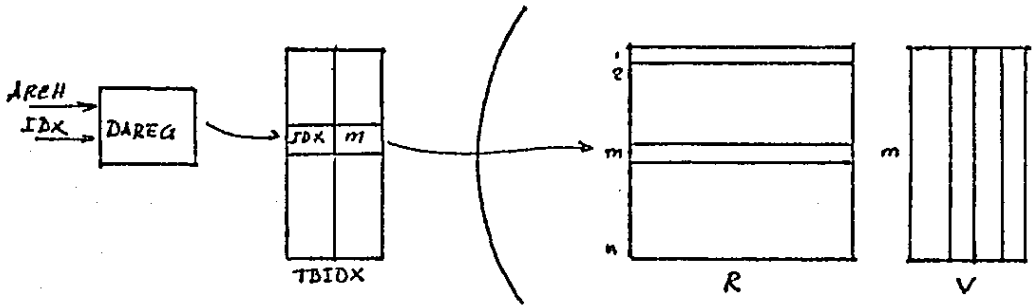
PROP - Nombre de la propiedad a consultar

VALP - Valor de la propiedad consultada (variable de salida)

V.1.1.1. Solución al acceso directo a registro.

Como ya fué mencionado en el capítulo de estructura física, este método de acceso se realiza a través de las tablas índices. Así, usando por un lado el identificador lógico (método de acceso del usuario) y por otro su correspondiente identificador interno (método de acceso del sistema) es posible obtener cualquier registro del archivo.

En la siguiente figura se ilustra este tipo de acceso.



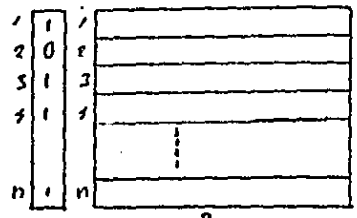
ARCH - archivo
TBIDX - tabla indice
DAREG - función que busca a IDX (Identificador logico)
m - localidad física donde se encuentra IDX (id. físico)

ACCESO DIRECTO A REGISTRO

FIG. V.1.1.1.A

La tabla índice es un archivo en disco, de tal forma que generalmente, toda operación de acceso por índice requiere, primero de una lectura de toda la tabla a memoria y posteriormente de una búsqueda, en memoria, del identificador; si hay restricciones en memoria principal para contener toda la tabla, esta se subdivide en tablas de tal manera que un acceso puede requerir de más de un acceso, primero de una subtabla que indicaría en que tabla puede estar el identificador buscado, un segundo acceso de esa tabla.

Vector de existencia. Para facilitar y sobre todo para dar un tiempo de respuesta rápido en las operaciones de bajas, se marcará cada registro como existente o no existente lógicamente, físicamente el registro sigue existiendo (en una operación de reorganización posterior, se darán las bajas físicas), de tal manera que toda operación tendrá que checar si el registro a operar existe. Así la existencia será una propiedad binaria (existe o no) de todo registro y se representará por un vector, el vector de existencia, donde un "1" indica que la propiedad existe y un "0" que no existe lógicamente, en la siguiente figura se ilustra dicha propiedad.



VECTOR DE EXISTENCIA

FIG. VI.1.2.1-B

En la figura se observa que el registro 2 no existe lógicamente, aunque físicamente está en R. En la operación bajas se verá la ventaja de esta propiedad de existencia.

V.1.2. Acceso por propiedad, o acceso vectorial

Con este método se accesará del archivo (o una parte de él) un conjunto de registros, lo cual resulta muy útil en operaciones donde se incluyen a todos los registros de un archivo o a una parte de él. La obtención de un subconjunto de registros se basará en una condición impuesta al total (predicado) o a una parte de un archivo.

Este acceso se expresa con la siguiente función:

CONSUL (ARCH,PRED,SALIDA,VEN,VSAL,PIDENTI)

ARCH - Nombre del archivo a accesar

PRED - Condición impuesta al conjunto de ARCH (predicado)

SALIDA - Especificación de que información se requiere de los registros que cumplieron la condición

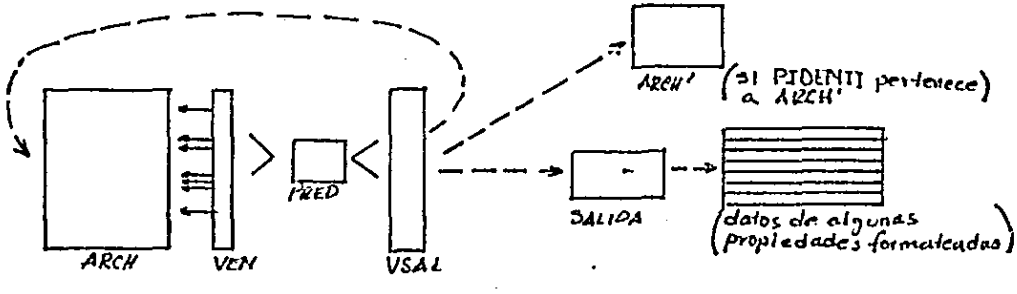
VEN - Vector de acceso de entrada, que especifica sobre que subconsjunto de ARCH, se va a aplicar el predicado.

VSAL - Vectores de acceso resultado, donde se dan las direcciones de los registros que cumplieron con el predicado.

PIDENTI - Nombre de la propiedad del identificador del archivo al cual va a apuntar VSAL. (este nombre de propiedad debe estar incluído en las propiedades de este registro; esto permitirá las operaciones combinadas).

E - Código de error en el caso que no se pueda realizar la operación.

Con la siguiente figura se ilustra este método de acceso.



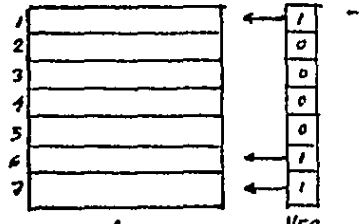
ACCESO POR PROPIEDAD (ACCESO VECTORIAL)

FIG. V.1.2.A

Dada un subconjunto (especificado por el vector de acceso VEN) de ARCH, se le aplica una cierta condición o restricción a algunas de las propiedades (PRED) de tal forma que se obtiene un nuevo subconjunto (especificador por el vector de acceso VSAL), el cual apuntará de acuerdo a la propiedad indentificar especificada (PIDENTI) que puede ser al mismo archivo o a otro cuyo identificador sea IPEDENTI. Del conjunto obtenido por VSAL, se formará la información a través de las especificaciones de la función SALIDA.

V.1.2.1 Solución al acceso por propiedad

Como también ya fué mencionado en el capítulo de estructura física, el acceso vectorial se realiza a través de un vector, que contiene "1"s y "0"s (igual al número de registros del archivo a que accesan) y que a través de los '1's y la posición que ocupan, apuntan a un conjunto de registros. En la siguiente figura se ilustra un vector de acceso:



VECTOR DE ACCESO

FIG. V 2.2.A

El vector (VEC) mostrado en la figura solo contempla del total de los registros de A, a tres, el 1, 6 y 7.

En el inciso de operaciones se verá ampliamente, la forma de obtención de los vectores de acceso a través de los predicados.

V.2. ACTUALIZACIONES

Las actualizaciones pueden ser de tres tipos:

Modificación o actualización, operación que consiste en cambiar uno o varios valores de propiedades de un registro en particular.

ALTAS

Operación que consiste en agregar un nuevo registro, con todas sus propiedades, a un archivo.

BAJAS

Operación inversa a la anterior, eliminación de un registro de un archivo.

V.2.1 Modificación

Esta operación se ejecuta sobre registros en particular, por lo que requiere del acceso a registros directos. Las operaciones básicas que se utilizan para esta operación son:

-acceso de lectura del registro

-operación sobre las propiedades

-acceso de escritura para actualización.

La operación modificación puede ser expresada con la siguiente función:

ACTUA (ARCH, PIDENTI=VAL, MODFS, E)

ARCH - Nombre del archivo

PIDENTI - Nombre del identificador de acceso

VAL - Valor del identificador del registro en particular al que va a modificar. Valores de propiedad.

MODFS - Arrego de dos dimensiones que contiene el conjunto de propiedades o cambios.

MODFS -(1)- Nombre de la propiedad

MODFS -(2)- Nuevo valor de la propiedad

E - Código de error, en el caso que no se haya podido realizar la operación.

V.2.1.1. Costo de operación de modificación

A Así el costo en cuanto accesos de esta operación es (donde en forma simplificada se considera el acceso a un elemento de la tabla índice - al de un registro - al de un vector - a "1" acceso).

TIPO	"accesos"
R	4
V	2nbp+2

nbp - no de bits en una propiedad

COSTOS EN MODIFICACION

FIG. V.2.1.1.A

Primero se requiere del acceso al vector de existencia (para checar si está el registro), segundo un acceso a la tabla de índices, después uno para el acceso del registro (recuperación) finalmente uno más, con algún valor de propiedad cambiado (actualización), dando un total de 4 accesos para esta operación en una estructura R.

Mientras que en una estructura V, donde una propiedad está representada por nbp vectores (registros físicos), donde nbp es el número de bits que se requieren para representar hasta el máximo valor posible de esa propiedad, el número de accesos requerido para la operación es 2xnbp, más el de la tabla de acceso y al de vectores de existencia dan un total de 2xnbp+2. Se puede observar que para este tipo de operación la estructuraRes más económica.

V.2.1.2 Solución a la operación de modificación.

El acceso, al registro se hará a través del acceso directo, es decir dado el identificador de registro. Sin embargo, dada las diferentes estructuras en las que puede estar representado el valor de una propiedad, la solución a la operación varía de acuerdo a ellas.

Solución en R.

Cuando la propiedad está en R o R&V, la operación resulta muy sencilla, se obtiene las propiedades del registro,

a través del acceso directo y se cambian las propiedades que se solicitan.

Solución en V

Si la propiedad está en V, se requerirán de tantos accesos, como vectores contiene la propiedad para ser representada. Se tomará de cada vector el bit indicado por el identificador interno, dado por el acceso directo, hasta formar totalmente el valor. En el algoritmo implementado realmente sólo se actualizan los vectores que cambian; por ejemplo supóngase que se tiene originalmente el valor de una propiedad en binario igual a 010010 y se desea cambiar a 110000 así con un primer acceso se lee el primer vector y a través de identificador interno (que dá la posición del bit) se ve su valor, si es diferente se cambia y con un segundo acceso se actualiza el vector, en caso contrario, no hay un segundo acceso, no se requiere la actualización porque el vector no ha cambiado. De esta forma el número de accesos para la operación de modificación (acceso de lectura y acceso de actualización) está entre nbp y 2nbp, el primer extremo cuando el valor a cambiar es exáctamente igual al original, y el segundo cuando es, binariamente, totalmente diferente (su complemento)

V.2.2. Altas

Al igual que la operación anterior, también esta operación se basa en el acceso a un registro en particular.

Esta operación puede ser expresada con la siguiente función:

ALTA (ARCH, NVLS, E)

ARCH - Nombre del archivo al cual se le va a dar de alta un registro.

NVLS - Arreglo que contiene los nombres de las propiedades así como sus valores.

NVLS(1)- nombre de propiedad

NVLS(2)- valor de propiedad

E - Código de error en el caso de que no se haya podido realizar la operación.

V.2.2.1 Costo de Operación de Alta.

Esta operación es similar a la de modificación con la diferencia que en esta se actualizan todos los valores de propiedad de un registro. Así el costo de esta operación se puede expresar como:

tipo	"accesos"
P	3
V	NV+2

COSTO EN ALTAS

NV- no total de vectores en V

FIG. V. 2.2.1.A

En R , se requiere un acceso para la alta en R, 1 para la alta a la tabla de índices y 1 más para el vector de existencia (ver bajas).

En V se reuieren del acceso de todos los vectores, 1 mas para la tabla de índices y 1 para el vector de existencia.

V.2.2.2. Solución a la operación Altas.

En este caso la dirección del nuevo registro, dado de alta, será n + 1 (donde n era el último registro en el archivo). Es claro que esta operación en V puede ser muy costosa, por la cantidad de vectores (= a 'accesos') que se requieren para formar todo un registro.

Por esta razón se puede dar una variante en la solución en el caso V. Si en la base de datos no se requiere de una actualización inmediata de las altas, el problema de un costo alto en tiempo de respuesta, puede ser resuelto con un archivo adicional (tipo R) que contiene areas de sobreflujo, donde una alta en V se haría realmente en este archivo (y no se tendrían que actualizar todos los vectores), con lo cual se tendría un costo económico para esta operación.

Un tiempo después y con un análisis de costos se tendría que realizar la integración de los datos a los archivos R y V. La consecuencia que está estructura puede ocasionar en las operaciones de la base son: toda operación en R o en V

tendrá que también analizar el área de sobreflujo, por ejemplo en una consulta global en V, después de la operación en V, tendrá que buscar en la tabla de sobreflujo, si algunos de los nuevos registros (aún no integrados a los archivos) también cumplen con el predicado, si es que se desea una respuesta muy actualizada o si no, solo dar la respuesta sin tomar en cuenta los últimas altas, esta variante no fué implementada en este sistema.

V.2.3 Bajas

Este tipo de operaciones, es muy similar a las operaciones de ALTAS, requiere del acceso directo a registros y en principio tendrá los mismos problemas en cueanto accesos.

La operación baja se puede expresar de la siguiente manera:

BAJA (ARCH, PIDENTI=VAL, E)

ARCH- Nombre del archivo

PIDENTI- Propiedad identificador

VAL- Valor de identificador (=al registro que se desea dar de baja).

E - Código con error en el caso en que no se haya podido llevar a cabo la operación.

V.2.3.1 Costo de operación de bajas.

En esta operación el costo es bajo no importando si esta en R o en V, simplemente el registro se marcará (seguirá existiendo físicamente) como dado de baja en el vector de existencia.

Posteriormente después de una evaluación de costos se llevará a cabo la eliminación física de los registros dados de baja, así el costo de esta operación es:

tipo	#accesos*
L	2
V	2

COSTO EN BAJAS

FIG. V. 2.3.1.A

El primer acceso es para leer la marca y cambiarla y el segundo es para actualizarla.

En el siguiente inciso se ampliará más esta operación V.2.3.2 Solución a la operación bajas.

Con el objeto de dar un tiempo de respuesta bueno, a todos los registros del archivo se les incluye una propiedad binaria (sólo dos valores) de existencia. De esta manera cuando un registro es dado de baja el registro se marca en ese vector. Así una baja (lógica y no física) requiere del acceso de lectura del vector de existencia y el acceso de su actualización

V.3 CONSULTAS

Esta operación permite conocer información de la base de datos, y puede ser de dos tipos.

- consulta a registro a través de identificador.
- consulta global a través de condición, en valores de propiedades, sobre todo el archivo.

V.3.1. Consulta o registro.

Para llevarse a cabo esta operación, se requiere del identificador lógico para accesar el registro requerido y conocer alguno de los valores de sus propiedades. Por lo que la operación requerirá del acceso directo.

La operación de consulta puede realizarse con las funciones DAREG (ARCH, PIDENTI = VAL, REG, E) y VALP= DAPROP (ARCH, PIDENTI=VAL, PROP, E), las cuales fueron explicadas en el inciso de acceso directo de registro (V.1.1)

V.3.2.1. Costo de operación para la operación consulta a un registro.

En la siguiente figura se ilustran los costos de "accesos"

tipo	"accesos"
R	3
V	nbp+2

nbp - no de bits en propiedad

COSTO EN CONSULTA A REGISTRO
FIG. V.3.2.1.A

En R, se consulta la tabla de índices, vector de existencia y finalmente al registro.

En V., además del de tablas índices y vector de existencia tendrá que accesarse todos los vectores de la propiedad.

V.3.2.2. Solución a la operación consulta a registro.

La solución a esta operación es similar a la solución de acceso directo, a través de la tabla índice se obtiene el identificador interno del registro lógico y partir de él se obtiene de forma directa, el valor de la propiedad si está en R y en forma indirecta (accesando todos los vectores de la propiedad) si está en V.

V.3.2. Consulta por predicados

Esta operación a diferencia de todas las anteriores, no se parte de los identificadores de los registros, si no se parte inversamente, dados ciertos valores de propiedad (condición a los registros del archivo) obtener quienes las cumplen. Así una parte fundamental de esta operación, es la condición impuesta, el predicado. Otra función importante dentro de esta operación es el formato de las propiedades que se desean conocer de los registros que cumplieron. A continuación se describe la función de consulta global:

CONSUL (ARCH, PRED, SALIDA, VEN, VSAL, PIDENTI, E).

la cual fué explicada en el inciso V.1.2., en bs siguientes incisos se ampliará más los conceptos de predicados y salidas.

V.3.2.1. PREDICADOS

Es una función lógica y/o aritmética y/o de comparación entre valores de propiedades y constantes, de tal manera que pueden expresar una condición de acceso impuesta a los registros de un archivo, por ejemplo saber quienes tienen edad=40 y son hombres, se puede expresar como un predicado aritmético y lógico EDAD=40 y SEXO=MASCULINO. En el inciso de operaciones se amplian más al respecto.

V.3.2.2. Salida

Generalmente en una consulta global no solo se desea saber quienes cumplen con el predicado, si no además conocer el valor de algunas de sus propiedades. Volviendo con el ejemplo del predicado anterior se desea saber de los que tienen edad=40 y son hombres, sus nombres y direcciones (no sólo su nombre que podría ser su identificador).

De esta manera una salida es una función que expresa que propiedades se desean imprimir y con qué formato se van a generar

V.3.2.2. Costo de operación de consulta por predicado

Dado que esta función está compuesta de dos más elementales, el costo se dividirá en dos partes, el costo de

consulta global producida por el predicado y el costo producido por la salida.

	TIPO	No. DE 'ACCESOS'
CONSULTA	R	n+1
	V	nbep + 1
SALIDA	R	VEC
	V	nbes

COSTO DE CONSULTAS POR PREDICADO

n- no. de regs. en el archivo
nbep- no de bits en predicado
nbes- no de bits en salida
VEC - no. de regs. que cumplen el predicado.

FIG. V.3.2.2-A

En la operación de consulta estando en R, dado que la consulta es sobre todo el archivo y sin hacer referencias a ningún identificador, es necesario recorrer todo el archivo, para resolver el predicado (n 'accesos') en cada uno de los registros, además del acceso al vector de existencia.

En el caso de una estructura V solo accesarán los vectores de las propiedades involucradas en el predicado (nbep), y su vector de existencia.

Es claro que en este tipo de operación la estructura óptima es la V. A diferencia de la estructura R, donde el tiempo de respuesta aumenta con la longitud del archivo, en una estructura V prácticamente es independiente, debido a que básicamente el tiempo de búsqueda en un disco (seek time) es mucho menor que el tiempo de transferencia.

Para el caso de la salida, una vez obtenidos el conjunto de registros que cumplen (=VEC) dados por un vector de acceso, el número de accesos que se requieren en estructura R, es igual a VEC, mientras que en estructura V , se requieren de tantos accesos como vectores contengan las propiedades implicadas en la salida (nbes).

Si la operación es sobre propiedades R, la solución es sencilla, una vez definido el predicado al accesar cada uno de los registros, se toman los valores de propiedades implicadas y se les aplica el predicado. Por ejemplo en un archivo de alumnos (estando en R) saber quienes son mujeres mayores de 22 años, al accesar cada registro se toman las propiedades sexo y edad y se aplica el predicado para saber si cumple o no.

Si la estructura es V, la operación cambia. Como se verá en el inciso operaciones, todo predicado puede reducirse a una función lógica vectorial, de tal manera que la solución de un predicado se realiza tomando cada vector de las propiedades y realizando operaciones lógicas entre ellas.

Finalmente una vez analizados los costos para las distintas operaciones, es claro que el acceso directo es adecuado a operaciones sobre registros y el acceso por conjuntos es adecuado a operaciones globales, así la estructura R resulta óptima cuando se opera por identificador y la estructura V resulta adecuada a las operaciones globales.

En la siguiente figura se ilustran juntos los resultados obtenidos:

MODIFICACION	R	4
	V	2nbp+2
ALTA	R	3
	V	nv+2
BAJA	R	2
	V	2
CONSULTA A REC.	R	3
	V	nbp+2
CONSULTA GLOBAL PREDICADO	R	n+1
	V	nbep+1
CONSULTA GLOBAL SALIDA	R	VEC
	V	nbes

COSTOS PARA OPERACIONES

FIG. II.3.2.2.B

V.4 OPERACIONES

Las operaciones que modifican los datos de las bases, son aquellas que afectan a los valores de las propiedades de los registros. Y dado la naturaleza del valor de una propiedad (valor lógico de existencia y valor numérico o alfabético) el tipo de operaciones que se le podrán realizar serán lógicas (y, o, NEG) de comparación ($=, \neq, >, <, \geq, \leq$) y aritméticas ($+, -, *, /$).

Las operaciones entre propiedades se pueden dividir en dos partes (definidas por los dos tipos de acceso), operación sobre propiedades de un registro y operaciones globales. (operaciones por predicado).

En los siguientes incisos se verán estos dos tipos de operaciones.

V.4.1. Operaciones sobre propiedades de un registro

Este tipo de operación se realiza utilizando el acceso directo, de tal forma que una vez obtenidos los valores de las propiedades podrán ser operados, usando las funciones lógicas, aritméticas y de comparación provistas por el lenguaje de programación con el que fué implementado del sistema.

V.4.2. Operaciones por predicado.

Este tipo de operaciones son las que permiten el acceso global (por vector) sobre un archivo.

Volviendo un poco a la implementación de una propiedad en (visto en el capítulo de estructura física) un archivo V a través de los vectores que lo forman. Supongase que se tiene el siguiente archivo con una representación de listas invertidas para sus propiedades

Nombre	edad	sexo	sueldo
JUAN	29	M	100
MARIA	19	F	200
PEDRO	19	M	100
ROSA	20	F	100
RAUL	21	M	100

identificador

edad	nombre
19	MARIA
	PEDRO
20	JUAN
	ROSA
21	RAUL

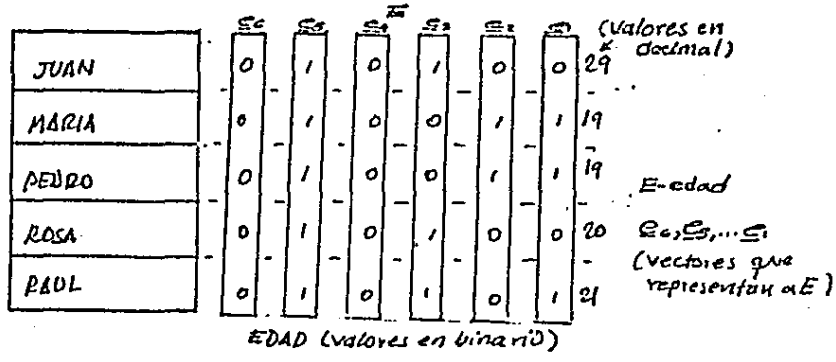
sexo	edad
M	JUAN
	PEDRO
	RAUL
F	MARIA
	ROSA

sueldo	edad
100	JUAN
	ROSA
	RAUL
150	PEDRO
200	MARIA

PROPIEDADES EN LISTAS INVERTIDAS

FIG. V.42.A.

Y ahora representemos, la propiedad edad a través de vectores (como fué explicado en el capítulo de estructura física), donde cada valor de propiedad es formado por su número binario correspondiente y donde cada bit junto con todos los de los demás registros forman un archivo. En la siguiente figura se implementa edad con vectores.



PROPIEDADES EN VECTORES

FIG. V.42.B

Con la anterior figura se observa que obtener los registros que cumplen con un cierto valor de propiedades consiste en operar lógicamente sus vectores y obtener un

vector de acceso que determinará quienes cumplen con el valor dado:

$$E = Val \quad \rightarrow \quad \underline{V}_{E=Val} = f(\underline{e}_i)$$

donde:

Val — un valor posible de la propiedad E, el cual se desea saber que registros la contienen.

$\underline{V}_{E=val}$ Vector de acceso que define que registros cumplen con la condición $E = val$.

f — Funcion logica entre vectores

$\underline{e}_i$ — Conjunto de vectores que componen a la propiedad E

Por ejemplo:

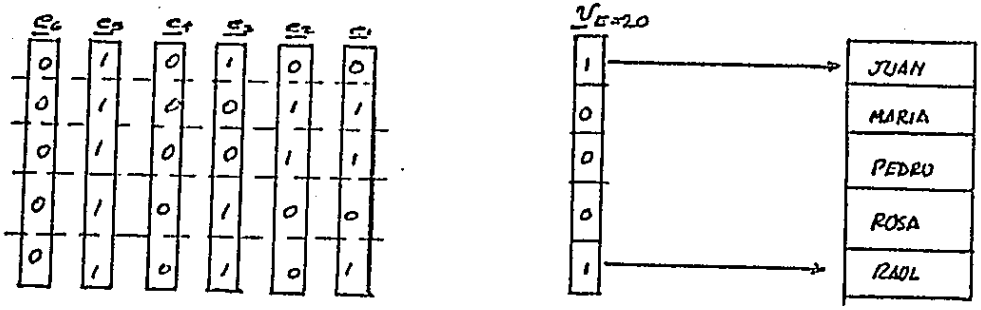
E=20 a (se requiere saber quienes tienen 20 años de edad).

$V_{E=20} = (\overline{e_6}, e_5, \overline{e_4}, e_3, \overline{e_2}, e_1)$

dado que 20 - 010100

y siguiendo la conversión de 0 = negado y

1 = cierto



VECTOR DE ACCESO
(donde 1 significa que el registro correspondiente cumple con E=20 y 0 no cumple)

OPERACIONES CON VECTORES

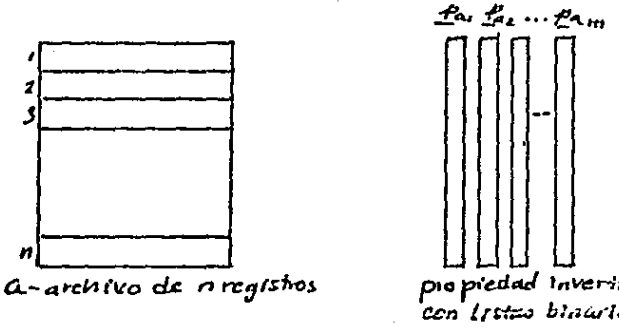
FIG. II-4.2.c.

Partiendo de que el archivo que se tiene es demasiado grandes y no es posible tenerlo en memoria principal , sólo en secundaria es fácil ver que una estructura de listas invertidas resulta eficiente en tiempo de respuesta para preguntas globales.

Haciendo referencia a la figura 1, saber quienes tienen

20 años consiste en consultar a Edad y buscar los que tienen 20, en lugar de buscar en todo el archivo y preguntar por cada registro quién tiene 20 años, en el caso que estuviera directo (no invertido)

Para nuestro caso de listas binarias invertidas, basta solo manipular unas cuantas listas (acceso de memoria secundaria de listas y operaciones lógicas) para tener una respuesta sin tener que recorrir a todo el archivo (en una estructura no invertida).



a- archivo de n registros

propiedad invertida con listas binarias

n = no. de registros en el archivo
m = no. de listas binarias (vectores) en la propiedad P
$(m = \text{entero}(\log_2(P_{max} - 1)) + 1)$
Pmax. maximo valor - entero, que puede alcanzar P

PROPIEDAD EN V

FIG. I. 4. 2. C

Suponiendo que ambas estructuras accesar a un registro o a una lista binaria representa un acceso físico, con la primera estructura se requiere de

n accesos
(n= no. de registros en el archivo)

mientras que para la segunda

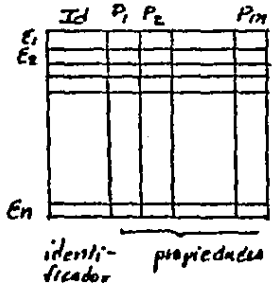
m accesos m es independiente de n (del tamaño
(m= bits en el predicado) del archivo).

y si n>> m, que es lo más factible,

este método por lo tanto es eficiente en tiempos de respuesta.

El manejo de listas binarias se generalizará definiendo las operaciones globales sobre archivos, pero antes se definirá de una manera más precisa a las listas binarias.

Dado un archivo como un conjunto de elementos (registros) diferentes (al menos por su identificador) con un conjunto de propiedades comunes que caracterizan a cada uno de ellos donde el valor de cúalquier propiedad asociado a cada uno de los elementos del archivo tendrá un valor entre 0 (no existencia de valor para la propiedad) y un valor máximo (vm)



conjunto de n elementos y m propiedades

$\forall P$ y $\forall E \in P$ = valor donde $0 \leq$ valor $\leq$ Vm

ARCHIVO LOGICO

FIG. V.4.2.D.

Se define a las listas binarias (vectores binarios) al conjunto de listas que se asocian a cada propiedad de tal forma que, el conjunto de listas binarias es igual, al número mínimo de bits para expresar el valor máximo (entero $\log_2$ vm - 1) y cada una de ellas con longitud n (igual al número de elementos del conjunto), con valores de 0 a 1, de modo que para cualquier valor de P asociado al elemento i (Ei) está representado por el valor binario dado por la concatenación de cada uno de los elementos del conjunto de vectores que representan dicha propiedad.

$$\forall P \in \underline{P_1}, \underline{P_2}, \ldots \underline{P_k}, \quad k = \text{entero}(\log_2 vm)+1$$

de tal forma que:

$$\forall P_i \in P_i = valor \equiv \underline{P_1}[i] \,\&\, \underline{P_2}[i] \,\&\, \ldots \underline{P_k}[i] \quad \& \text{ concatenación}$$

(valor binario de P)

Ahora difiniendo como operaciones globales sobre todo el archivo, como la operación entre 1 o varias propiedades para cada uno de los elementos del archivo:

dado

$$P_{isol} = P_{i1}\,op\,P_{i2}\,op \ldots P_{in} \qquad i=1,2,\ldots n$$

(operación global sobre todo el archivo)

P_{isol} = resultado de la operación global, sobre el elem i, entre las propiedades

op = operadores (lógicos, de relación o numéricos)

Dado (1) existe una función binaria entre los vectores de las propiedades involucrads en la operación tal que:

$$P_{sol} = \underline{p}_{sol_1}, \underline{p}_{sol_2}, \cdots \underline{p}_{solf}$$

donde

$$\underline{p}_{sol_1}, \underline{p}_{sol_2} \cdots \underline{p}_{solf} = \mathcal{F}(\underline{p}_1, \underline{p}_{a_2}, \cdots \underline{p}_{a_g}, \underline{p}_{b_1}, \underline{p}_{b_2} \cdots \underline{p}_{b_h}, \cdots \underline{p}_{z_1}, \underline{p}_{z_2} \cdots \underline{p}_{z_f})$$

$\underline{p}$ vectores que forman a la propiedad p

f función lógica vectorial

donde

g entero$(log_2 Vm_a)+1$

h entero $(log_2 Vm_b)+1$

⋮

f = número de vectores, que pueden representar la propiedad y dependerá del tipo de tensión lógica a expresar

$\mathcal{F}$ = función lógica en vectores binarios (= a la operación lógica entre cada dnco de los elementos de los vectores)

$$\left(\mathcal{F}(\underline{a}, \underline{b}) = \underset{i=1,2,\cdots n}{f(\underline{a}[i], \underline{b}[i])} = \underline{a}[i] \text{ op } \underline{b}[i] \right.$$

a y b vectores binarios

$\mathcal{F}$ = función lógica entre vectores

f = función lógica

op = operando lógico (y, o, no)

Es decir el problema de operar globalmente un archivo (1) se reduce a resolver lógicamente el conjunto de vectores de las propiedades involucradas.

La ventaja que ofrece obtener una solución (1) indirectamente a través de (2) es resolver una operación global (lógica, relación o numérica) con un mínimo de accesos sin tener que recurrir a todo el archivo.

Supongamos que se requiere obtener un total de diferentes propiedades para todo el archivo, por ejemplo, cantidad total de artículos vendidos por vendedor, operación que consistirá en sumar para cada registro de vendedor la cantidad de artículos A vendidos, de artículos B vendidos, etc. y además, partiendo de que toda la información está en disco y que obtener un registro o un vector binario sgnifica un acceso a disco, se observa que la cantidad de accesos a través de (2) es independiente de los elementos del archivo y depende del número de listas binarias que contengan las propiedades involucradas. Suponiendo para el ejemplo anterior, que existen 10 productos y que para cada uno de ellos se requieren 10 listas ($2^{10}=1024$ diferentes valores) se podría esperar que se necesitará 100 accesos (10 vectores x 10 propiedades, que no necesariamentes es exacto, pueden ser menos, o más dependiendo de la operación y de la forma expresada por (2)).

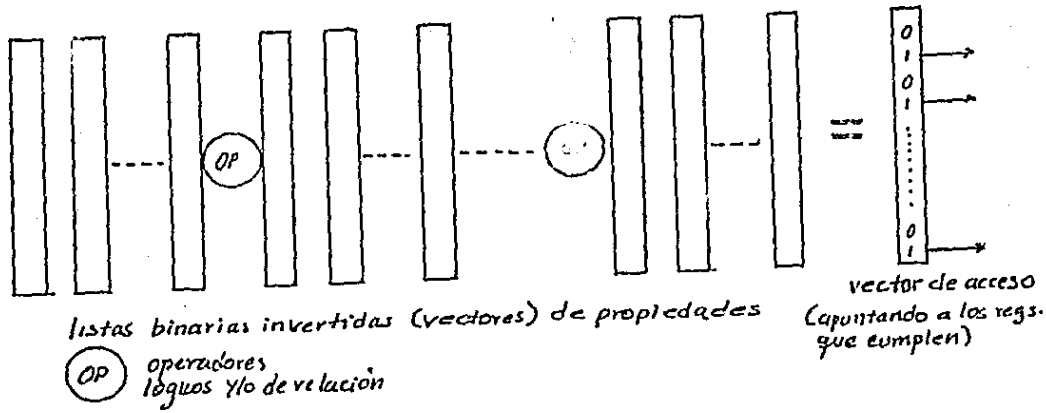
Si el archivo consta de 1000 registros (un archivo pequeño) la solución por función binaria de vectores (2), resulta ser 10 veces más rápido en accesos, en contra de una solución secuencial de registro por registros(1000 accesos).

De esta forma el problema a resolver consiste en expresar cualquier operación global (1) (Lógica, relación o numérica) de la forma (2) (lógica, vectorial binaria).

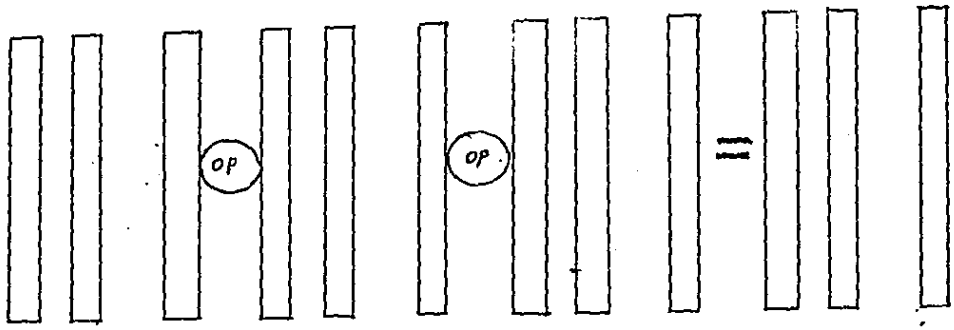
Los algoritmos para obtener la función lógica vectorial binaria, se pueden dividir en dos en cuanto al tipo de operación global a resolver.

Cuando los operadores son lógicos y/o de relación, la solución es binaria en el sentido de que el registro simplemente cumple o no con la condición impuesta, por lo tanto en la solución solo existe en lista binaria ($P_{sol} \rightarrow \underline{P}_{sol}$) que se define como vector de acceso, el cual determina quienes cumplen con la operación requerida o no, con un 1 o un 0 según sea la convensión, por ejemplo edad=20 años ¿quienes son hombres y casados? (lógica), la respuesta es si o no, cumple la condición.

Cuando los operadores son numéricos la solución no es una sola lista sino un conjunto de listas, que contienen el resultado numérico en binario de la operación, por ejemplo el total de productos vendidos.



VECTOR DE ACCESO



VECTORES DE SOLUCION

FIG. II.1.2 E.

Operaciones

a) lógicas

1) Pa , $\overline{pa}$ (existencia o inexistencia del valor de la propiedad).

2) $Pa \cdot pb$ (y lógico entre dos propiedades)

3) $Pa + pb$ (o lógico entre dos propiedades)

b) relación

1) $pa = pb$ 1') $Pa = c$

2) $Pa \neq pb$ 2') $pa \neq cte$

3) $Pa > pb$ 3') $pa > cte$

4) $Pa \leq pb$ 4') $pa \leq cte$

5) $Pa < pb$ 5') $Pa < cte$

6) $Pa \geq pb$ 6') $Pa \geq cte$

7') $cte_1 \leq pa \leq cte_2$

c) numéricos

1) $Pa + pb$ suma entre dos propiedades

2) $Pa - pb$ resta entre dos propiedades

3) $Pa * pb$ productos entre dos propiedades

4) pa/pb división entre dos propiedades

V.4.2.1. Algoritmos para las operaciones globales.

Las características de las propiedades en V y las operaciones globales pueden ser resumidas como sigue:

Toda propiedad de un archivo V está formado por un conjunto de vectores que la representan

$$\forall P \in \underline{P} = \underline{p}_1, \underline{p}_2, \underline{p}_3 \cdots \underline{p}_n \quad \text{—— (1)}$$

donde p - propiedad

$\underline{p}$ - conjunto de vectores que representan a P

$\underline{p}i$ - el vector i de P

n - no de vectores necesarios, para representar a P

n = entero $(\log_2 \text{max.val.})$.

Toda operación condicional puede ser expresada por vector de acceso que es una función lógica de los vectores de las propiedades implicadas.

$$(P_a, P_b, \cdots P_m) \equiv \underline{V}_{(P_a, P_b, \ldots P_n)} = \int (\underline{P}_a, \underline{P}_b, \cdots \underline{P}_m) \quad \text{—— (2)}$$

donde

$(P_a, P_b, \ldots P_m)$ operación condiciona

(lógica y/o relación) entre las propiedades $P_a, P_b \ldots P_m$

m - ultima propiedad implicada en la condicion

Toda operación aritmética entre propiedades darán como resultado a un conjunto de vectores los cuales son función de los vectores de las propiedades implicadas:

$$S = (P_a, P_b, \ldots P_m) = f(\underline{P_a}, \underline{P_b} \ldots \underline{P_m}) \quad \text{—— (3)}$$

donde

S- es la solución a la operación aritmética entre las propiedades $P_a, P_b, \ldots P_m$

$(P_a, P_b, \ldots P_m)$- operación aritmética entre propiedades

A continuación se dan los algoritmos de solución para cada una de las operaciones definidas, antes solo hay que hacer notar que las operaciones con cada uno de los valores de un bit de una propiedad son realmente todos los valores del mismo bit de la misma propiedad de todos los registros del archivo que están representados por un vector, por lo tanto no es posible conocer el valor lógico del vector.

En las operaciones que incluyen más de una propiedad, se considera que todas las propiedades cuentan con n vectores que la representan, si no fuese así n será igual al máximo de vectores de la propiedad con mayor número de ellos y el resto de las propiedades se le añadirán vectores $\underline{0}$, en los bits más significativos sin influir por esto en el resultado.

Se define a $\underline{0}$ como un sector cuyos elementos son 0s y a $\underline{1}$ como un vector cuyos elementos son 1s.

En las operaciones aritméticas el resultado entre dos propiedades tendrá un número diferente de n, en la suma y resta n+1, en el producto 2xn y en la división n. Finalmente en las operaciones se considerarán propiedades enteras positivas, no se incluyen numeros reales ni de punto flotante.

a) Lógicas

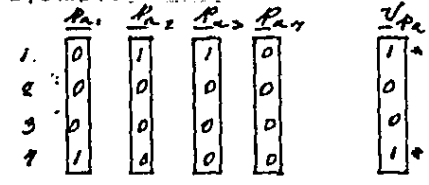
1.- Existencia de la propiedad (pa) y la función negada de la propiedad a ($\bar{p}a$).

$$\underline{Pa} \equiv \underline{\bigcup} \underline{P_a} = \underline{p_{cu}} + \underline{p_{a_2}} \underline{1} \underline{p_{a_1}} \underline{1} \cdots \underline{p_{an}} \quad -- (i)$$
$$\underline{\bar{P}a} \equiv \underline{\bigvee} \underline{\bar{P}_a} = \underline{\bar{V}_{Pa}}$$

Se definirá que una propiedad existe cuando tiene un valor diferente de 0 y no existe cuando tiene el valor 0.

De esta manera haciendo el '0' lógico con cada uno de los bits que componen su valor, al final se tiene el valor, existe o no, en el vector de acceso.

Ejemplo



EXISTENCIA DE UNA PROPIEDAD

FIG-V.4.2.1.A-

2.- Y lógico de dos propiedades (Pa · Pb)

$$P_a \cdot P_b \equiv \underline{V}_{P_a \cdot P_b} = \underline{V}_{P_a} \cdot \underline{V}_{P_b}$$

Esta operación lógica nos dará un valor cierto (= 1) en el vector de acceso, cuando existan valores tanto en Pa como en Pb. De esta forma se aplica (i) para ambas propiedades, y a los vectores de acceso obtenido se les operará con un y lógico, para así finalmente tener el vector de acceso $\underline{V}_{a\ P_a \cdot P_b}$

3.- O Lógico de dos propiedades (Pa + Pb)

$$P_a + P_b \equiv \underline{V}_{P_a + P_b} = \underline{V}_{P_a} + \underline{V}_{P_b}$$

Esta operación lógica es similar a la anterior excepto que se realiza una operación 'O' lógica entre los vectores de acceso.

Con estas operaciones lógicas puede realizarse cualquier función lógica (por ejemplo el 'O' exclusivo) ⊕

$$P_a \oplus P_b \equiv \underline{V}_{P_a \oplus P_b} = (\underline{V}_{P_a} \cdot \overline{\underline{V}}_{P_b}) + (\overline{\underline{V}}_{P_a} \cdot \underline{V}_{P_b})$$

b) Relación

1.- Igualdad de valores de propiedades (Pa = Pb)

$$P_a = P_b \equiv \underline{V}_{a\ P_a = P_b} = \overline{\underline{P}_{a_1} \oplus \underline{P}_{b_1}} \cdot \overline{\underline{P}_{a_2} \oplus \underline{P}_{b_2}} \cdot \cdots \overline{\underline{P}_{a_n} \oplus \underline{P}_{b_n}} \quad (ii)$$

El O exclusivo (⊕) negado nos da como valor cierto, cuando dos valores son iguales, de tal forma que aplicando un 'y' lógico al 'O' exclusivo negado de cada par de valores correspondientes, al final se tendrá en el vector de acceso el valor 1 si todos los bits de ambas propiedades son iguales

en caso contrario da 0.

2.- Diferencia entre propiedades (Pa ≠ Pb)

$$Pa \neq Pb := \underline{V}_{Pa \neq Pb} = \overline{\underline{V}}_{Pa=Pb}$$

Esta operación puede ser obtenida con la expresión anterior y negando el vector de acceso.

3.- Propiedad mayor que otra (Pa > Pb)

Donde $\underline{\Delta_3}$ se obtiene así:

para i = 1, 2, ----n donde $\underline{s}_1$ inicial $=\underline{0}$ y $\underline{s}_2$ inicial $=\underline{0}$
$\underline{s}_3$ inicial $= 0$

$$\underline{\Delta_1} = \underline{\Delta_1} + (\underline{P}_{ai} \cdot \underline{P}_{bi})$$
$$\underline{\Delta_2} = \underline{\Delta_2} + (\overline{\underline{P}}_{ai} \cdot \underline{P}_{bi})$$
$$\underline{\Delta_3} = \underline{\Delta_3} + (\underline{\Delta_1} \cdot \overline{\underline{\Delta_2}}) \quad \text{——— (iii)}$$

Lo que determina que en dos valores en binario uno sea mayor que otro (por ejemplo $a = 0_1 1_2 0_3 0_4 1_5 0_6$ y $b = 0_1 1_2 0_3 0_4 0_5 0_6$) es que en la primera diferencia de pares de bits correspondientes (de i= 1, 2, --- 6 en este ejemplo) entre las dos cantidades, el bit correspondiente (bit no 5, en a=1 y en b=0) el de a sea 1 y el de b sea 0, el resto no importa pues basta el más significante. El caso contrario, donde el bit de b=1 y el de a=0 significa que b > a.

De esta forma observando las expresiones (iii)

$\underline{s}_1$ quedará "prendido" si la primera diferencia es "1" y

pero si la diferencia es "0" y "1" será $\underline{S}2$ la que se quedará prendido, de tal forma que si es S_1 la que se prende primero, (S_2 no puede prenderse al mismo tiempo) $S3$ quedará prendido indicando que Pa>pb, pero si es $\underline{S}2$ la que se prendió primero, $\underline{S}3$ nunca podrá prenderse porque S2 está negado en la expresión $\underline{S}3=\underline{S}3+(\underline{S}1 . \overline{\underline{S}}2)$. En realidad decir prender al vector $\underline{S}$ no significa que $\underline{S}=\underline{1}$, ya que dentro del mismo vector $\underline{S}$ algunos bits se prenden y otros no con el siguiente ejemplo quedará más claro el algoritmo.

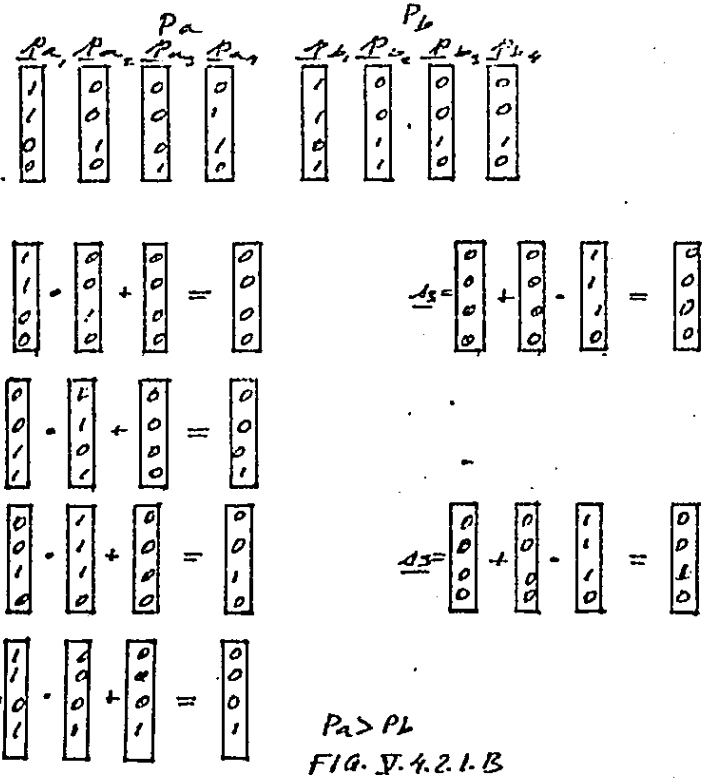


FIG. V.4.2.1.B

$$\underline{\Delta_1} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix} \cdot \begin{bmatrix}1\\1\\0\\1\end{bmatrix} + \begin{bmatrix}0\\0\\1\\0\end{bmatrix} = \begin{bmatrix}0\\0\\1\\1\end{bmatrix} \qquad \underline{\Delta_3} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix} + \begin{bmatrix}0\\0\\1\\1\end{bmatrix} \cdot \begin{bmatrix}1\\1\\0\\0\end{bmatrix} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$$

$$\underline{\Delta_2} = \begin{bmatrix}1\\1\\1\\0\end{bmatrix} \cdot \begin{bmatrix}0\\0\\1\\0\end{bmatrix} + \begin{bmatrix}0\\0\\0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

$$\underline{\Delta_1} = \begin{bmatrix}0\\1\\1\\0\end{bmatrix} \cdot \begin{bmatrix}1\\1\\0\\1\end{bmatrix} + \begin{bmatrix}0\\0\\1\\1\end{bmatrix} = \begin{bmatrix}0\\1\\1\\1\end{bmatrix} \qquad \underline{\Delta_3} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix} + \begin{bmatrix}0\\1\\1\\1\end{bmatrix} \cdot \begin{bmatrix}1\\1\\0\\0\end{bmatrix} = \begin{bmatrix}0\\1\\1\\0\end{bmatrix}$$

$$\underline{\Delta_2} = \begin{bmatrix}1\\0\\0\\1\end{bmatrix} \cdot \begin{bmatrix}0\\0\\1\\0\end{bmatrix} + \begin{bmatrix}0\\0\\1\\1\end{bmatrix} = \begin{bmatrix}0\\0\\1\\1\end{bmatrix}$$

El vector S3 da la solución

1	0
2	1
3	1
4	0

$P_{a1} \leq P_{b1}$

$P_{a2} \geq P_{b2}$

$P_{a3} > P_{b3}$

$P_{a4} \leq P_{b4}$

haciendo la comparación por renglón

	P_a		P_b		
1	1 0 0 0	es >	1 0 0 0	no	(0)
2	1 0 0 0	es >	1 0 0 1	si	(1)
3	0 1 0 1	es >	0 0 1 1	no	(1)
4	0 0 1 0	es >	1 1 0 0	si	(0)

$\equiv \begin{bmatrix}0\\1\\1\\0\end{bmatrix} = \underline{\Delta_3}$

$P_a > P_b$

FIG. V.4.2.1.B

4.- Propiedad menor igual que otra Pa ≤ Pb.

$$Pa \leq Pb \equiv \underline{V}_{Pa \leq Pb} = \overline{V}_{Pa > Pb}$$

Para este caso se utilizan las expresiones (iii) y se invierte el vector de acceso.

5.- Propiedad Pa < Pb

$$Pa < Pb \equiv \overline{V}_{Pa < Pb} = \underline{V}_{Pb > Pa}$$

En este caso también se usa (iii) y solo se cambia el orden de las propiedades.

6.- Propiedad Pa ≥ Pb

$$Pa \geq Pb \equiv \underline{V}_{Pa \geq Pb} = \overline{V}_{Pa < Pb}$$

También se usa (iii)

1'.- Propiedad igual a una constante

Pa= cte donde cte expresado en binarios es: $b_1 \& b_2 \& \ldots \& b_n$

$$\underline{V}_{Pa=CTE} = \underline{P}'_{a1} \underline{P}'_{a2} \underline{P}_{a3} \cdots \underline{P}'_{an} \quad \text{—— (iv)}$$

donde

$$\underline{P}'_{ai} = \overline{P}_{ai} \text{ si } b_i = 0$$

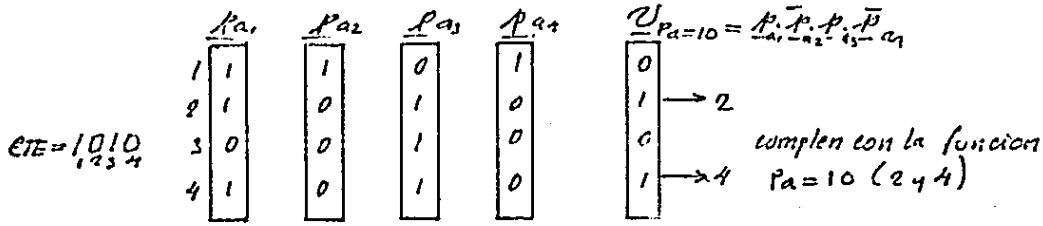
y

$$\underline{P}'_{ai} = \underline{P}_{ai} \text{ si } b_i = 1$$

Si de la constante en binario se hace una función lógica considerando sus ceros como la negación y los unos como su afirmación al aplicarles un 'y' lógico al final se

tiene que el vector de acceso solo apunte aquellos que tengan el valor de la constante. Por ejemplo: $P_A = 1010$

$\underline{V}_{Pa=10} = \underline{P}_{a_1} \cdot \overline{\underline{P}}_{a_2} \cdot \underline{P}_{a_3} \cdot \overline{\underline{P}}_{a_4}$ supongamos que cada vector está expresado como sigue:



2'.- Propiedad diferente a constante

$$P \neq CTE \equiv \underline{V}_{P \neq CTE} = \overline{\underline{V}}_{P=CTE}$$

Para este caso se usa la expresión anterior (iv) y se niega : al vector de acc[illegible].

3'.- Propiedad mayor o igual que una constante

$P \geq CTE \equiv \underline{V}_{P \geq CTE} = \underline{S}$, $CTE = b_1 \, 8 \, b_2 \, 8 \cdots 8 \, b_n$

donde $\underline{S}$ se obtiene y dando $\underline{S}$ incial = $\underline{1}$
para $i = n, n-1, n-2, \ldots$

si $b_i = 0$ $\underline{S} = \underline{S} + \overline{\underline{P}}_i$ —— (v)
si $b_i = 1$ $\underline{S} = \underline{S} \cdot \underline{P}_i$

Para que un número en binario sea mayor o igual que una constante debe tener al menos los mismos 1s que los de la constante y en la misma posición, si son iguales tendrán en la misma posición los 1s y los ceros y si es mayor tendrá los 1s correspondientes a los de las constantes y otros más que nos indicarían que la propiedad es mayor que la constante. Así, por ejemplo, la constante CTE=01010

CTE=01010

un número conservando al menos los mismos 1 y en la misma posición es

N=x1x1x, donde x puede ser 1 ó 0

Será = o mayor. De esta forma lo importante en la función es que cuando se detecte un '1' en la constante.es importante que la función se cumpla con seguridad (para poder saber si es igual o mayor) así $\underline{S}=\underline{S} \bullet \underline{b}i$ y si es un "0" no importa (puede ser igual si no cumple o mayor si cumple) así $\underline{S}=\underline{S}+\underline{b}i$. Para explicar más sobre el caso supóngase que CTE=1010

4'.- Propiedad mayor que una constante

$$Pa > CTE = \underline{V}_{Pa>CTE} = \underline{U}_{Pa \geq CTE'},\ CTE' = CTE+1$$

En esta operación se usa la expresión (v) sumandole 1 a la constante.

5'.- Propiedad menor o igual que una constante

$$P_a \leq CTE \equiv \underline{V}_{P_a \leq CTE} \equiv \overline{V}_{P_a > CTE}$$

Esta expresión se obtiene invirtiendo lo anterior

6'.- Propiedad menor que constante

$$P_a < CTE \equiv \underline{V}_{P < CTE} \equiv \overline{V}_{P \geq CTE}$$

Esta expresión se obtiene negando a la expresión (v)

7.- Propiedad entre constantes

$$CTE_1 \leq P_a \leq CTE_2 \quad \underline{V}_{CTE_1 \leq P_a \leq CTE_2} = f_1 + f_2$$
$$f_1 = \underline{V}_{P_a \geq CTE_1}, \quad f_2 = \underline{V}_{P_a \leq CTE_2}$$

Esta expresión es una combinación de las anteriores.

c) Operaciones numéricas. En este tipo de operaciones es importante señalar que a diferencia de las operaciones anteriores el resultado no será un vector de acceso sino un conjunto de vectores que expresará el resultado de la operación.

1.- Suma de dos propiedades

$$P_S = P_a + P_b \equiv \underline{P}_S = f(\underline{P}_a, \underline{P}_b) = \underline{a}_0, \underline{s}_1, \underline{s}_2, \ldots \underline{s}_n = \underline{a}_0, \underline{S}$$

para $i = n, n-1, n-2, \cdots 1$ y $\underline{a}_n = \underline{0}$

$$\underline{r}_i = (P_{a_i} \oplus P_{b_i})$$
$$\underline{s}_i = (\underline{r}_i \oplus \underline{a}_i)$$
$$\underline{a}_{i-1} = (P_{a_i} \cdot P_{b_i}) \oplus (\underline{r}_i \cdot \underline{s}_i) \quad \text{——— (vi)}$$

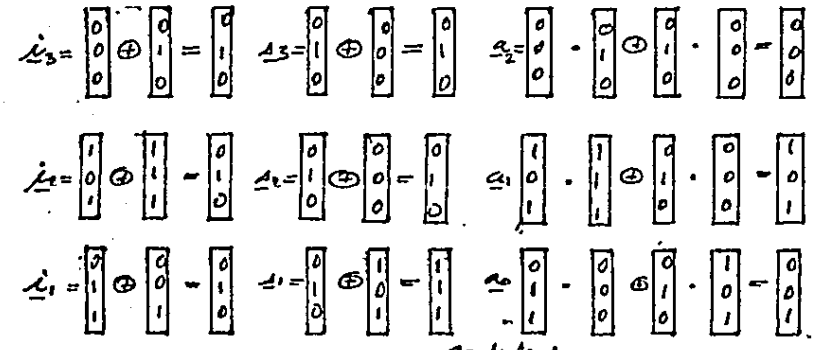
Esta operación dará un conjunto de vectores ($\underline{a}_0, \underline{S}$) que representará la suma de dos propiedades también expresadas en vectores ($\underline{P}_a, \underline{P}_b$).

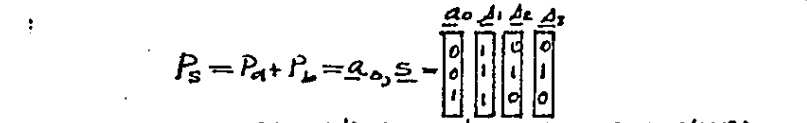
Primero se suman los vectores menos significantes ($\underline{s}_i$) posteriormente se le suman el acarreo ($\underline{A}_i$) y finalmente se calcula el acarreo de esta suma ($\underline{a}_{i-1}$).

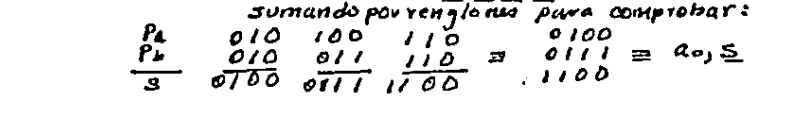
Es de observarse que el resultado tendrá un vector más que N, que es el último acarreo de la suma.

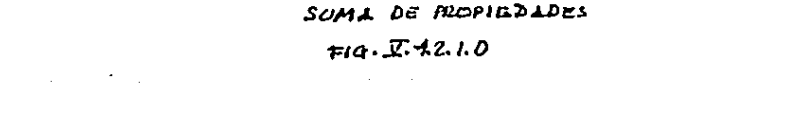
A continuación se dá un ejemplo.

$\underline{a}_3 = \underline{0}$









$P_S = P_a + P_b = \underline{a}_0, \underline{S}$

Sumando por renglones para comprobar:

P_a	010	100	110	0100
P_b	010	011	110	0111 ≡ $\underline{a}_0, \underline{S}$
S	0100	0111	1100	1100

SUMA DE PROPIEDADES

FIG. V.42.1.0

2.- Diferencia entre propiedades

$$P_D = P_a - P_b \equiv \underline{P_D} = f(\underline{P_a}, \underline{P_b}) = \underline{a_0}, \underline{d_1}, \underline{d_2}, \ldots \underline{d_n} = \underline{a_0}, \underline{D}$$

para $i = n, n-1, n-2, \ldots 1$ y $\underline{a}\, n = \underline{0}$

$$\underline{i_i} = (\underline{p_{ai}} \oplus \underline{p_{bi}})$$

$$\underline{d_i} = \overline{\underline{i_i}} \oplus \underline{a_i}$$

$$\underline{a_{i-1}} = (\underline{p_{ai}} \cdot \underline{p_{bi}}) \oplus (\overline{\underline{i_i}} \cdot \underline{a_i}) \quad \text{(viii)}$$

Este caso es muy similar a la anterior excepto que la acumulación es negativa, al final del resultado será un conjunto de vectores (n + 1) donde $\underline{a_0}$ dará el signo de la diferencia si es 0 el resultado es positivo en caso contrario será negativo.

3.- Producto entre propiedades

$$P_p = P_a * P_b \equiv \underline{P_p} = f(\underline{P_a}, \underline{P_b})$$

este algoritmo no fué desarrollado

4.- División entre propiedades Pa/pb

$$P_c = P_a / P_b \equiv \underline{P_c} = f(\underline{P_a}, \underline{P_b})$$

este algoritmo tampoco fué desarrollado

1'.- Suma con una constante Pa+CTE

$$P_S = P_a + CTE \equiv \underline{P_S} = f(\underline{P_a}, CTE) = \underline{a}\,, \quad \underline{S} = \underline{a_0}, \underline{s_1}, \underline{s_2}, \cdots \underline{s_n}$$

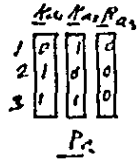
para $i = n, n-1, n-2, \ldots 1$ $\underline{a} = \underline{0}$

$CTE = c_1 c_2 c_3 \ldots c_i$

si $c_i = 0$ | $\underline{s_i} = \underline{p_{ai}} \oplus \underline{a}$; $\underline{a} = \underline{p_{ai}} \cdot \underline{a}$

si $c_i = 1$ | $\underline{s_i} = \overline{\underline{p_{ai}}} \oplus \underline{a}$; $\underline{a} = \underline{p_{ai}} + \overline{\underline{p_{ai}}} \cdot \underline{a}$

En este caso a diferencia de la suma de propiedades resulta ser más sencillo, ya que si se puede conocer los bits de la constante, de tal manera que dependiendo del bit de la constante se hace primero la suma del acarreo (si) y posterirormente se calcula el acarreo. A continuación se da un ejemplo.



$CTE = 010$ $\quad c_1 = 0,\ c_2 = 1,\ c_3 = 0$ $\quad \underline{a} = \begin{bmatrix}0\\0\\0\end{bmatrix} = \underline{0}$

$c_3 = 0$ $\quad \underline{A}_3 = \begin{bmatrix}0\\0\\0\end{bmatrix} \oplus \begin{bmatrix}0\\0\\0\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}$ $\quad \underline{a} = \begin{bmatrix}0\\0\\0\end{bmatrix} \cdot \begin{bmatrix}0\\0\\0\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}$

$c_2 = 0$ $\quad \underline{A}_2 = \begin{bmatrix}0\\0\\0\end{bmatrix} \oplus \begin{bmatrix}0\\0\\0\end{bmatrix} = \begin{bmatrix}0\\1\\0\end{bmatrix}$ $\quad \underline{a} = \begin{bmatrix}1\\0\\1\end{bmatrix} \cdot \begin{bmatrix}0\\0\\0\end{bmatrix} = \begin{bmatrix}1\\0\\1\end{bmatrix}$

$c_1 = 0$ $\quad \underline{A}_1 = \begin{bmatrix}0\\1\\1\end{bmatrix} \oplus \begin{bmatrix}1\\0\\1\end{bmatrix} = \begin{bmatrix}1\\1\\0\end{bmatrix}$ $\quad \underline{a} = \begin{bmatrix}0\\1\\1\end{bmatrix} \cdot \begin{bmatrix}1\\0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\1\end{bmatrix}$

$\underline{a}_1 \underline{S}_1 \underline{A}_2 \underline{A}_3 \begin{bmatrix}0\\0\\1\end{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix}\begin{bmatrix}0\\1\\0\end{bmatrix}\begin{bmatrix}0\\0\\0\end{bmatrix} \equiv P_3 = P_2 + CTE = \underline{a}_0, \underline{S}$

sumando por renglones para comprobar:

P_2	010	100	100	0100
CTE	010	010	100	0110 $\equiv \underline{a}_1 \underline{S}$
	0100	0110	1000	1000

2.- Diferencia de una propiedad con una constante

$$P_S = P_a - CTE \Rightarrow \underline{P}_S = f(\underline{P}_a, CTE) = \underline{a}, \underline{S} = a_1, a_2, \ldots, a_n$$

para $i = n, n-1, n-2, \ldots 1$ $\underline{a}$ inicial $= \underline{0}$ y $CTE = a_1 a_2 a_3 \ldots$

$$\text{si } e_i = 0 \;/\; \begin{array}{l} \underline{S}_i = \underline{P}_{ai} \oplus \underline{a} \\ \underline{a} = \underline{P}_{ai} \cdot \underline{a} \end{array}$$

$$\text{si } e_i = 1 \;/\; \begin{array}{l} \underline{S}_i = \underline{P}_{ai} \oplus \underline{a} \\ \underline{a} = \underline{P}_{ai} + \underline{P}_{ai} \cdot \underline{a} \end{array} \quad \text{(xii)}$$

Esta expresión es muy similar a la anterior, solo hay que tomar en cuenta que el acarreo es negativo y que los bits del vector $\underline{a}$, al final darán el signo del resultado de la diferencia (0 = + y 1 = −)

A continuación se dá un ejemplo que ilustra esta expresión:

$$\begin{matrix}1\\2\\3\end{matrix}\begin{bmatrix}1\\0\\1\end{bmatrix}\begin{bmatrix}a_1\\a_2\\a_3\end{bmatrix}\begin{bmatrix}a\\1\\2\end{bmatrix} \qquad CTE = 010 \qquad \begin{matrix}c_1 = 0\\c_2 = 1\\c_3 = 0\end{matrix} \qquad \underline{S} = \underline{\alpha} = \begin{bmatrix}0\\0\\0\end{bmatrix}$$

$$\underline{A_3} = \begin{bmatrix}0\\1\\0\end{bmatrix} \oplus \begin{bmatrix}a\\0\\0\end{bmatrix} \begin{bmatrix}a\\1\\0\end{bmatrix} \qquad \underline{\alpha} = \begin{bmatrix}1\\0\\1\end{bmatrix} \cdot \begin{bmatrix}0\\0\\0\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}$$

$$\underline{A_2} = \begin{bmatrix}1\\1\\0\end{bmatrix} \oplus \begin{bmatrix}\\0\\0\end{bmatrix} = \begin{bmatrix}1\\1\\0\end{bmatrix} \qquad \underline{\alpha} = \begin{bmatrix}1\\1\\0\end{bmatrix} \cdot \begin{bmatrix}\\0\\1\end{bmatrix} = \begin{bmatrix}1\\1\\0\end{bmatrix}$$

$$\underline{A_1} = \begin{bmatrix}1\\0\\1\end{bmatrix} \oplus \begin{bmatrix}1\\1\\0\end{bmatrix} = \begin{bmatrix}0\\1\\1\end{bmatrix} \qquad \underline{\alpha} = \begin{bmatrix}0\\1\\0\end{bmatrix} \cdot \begin{bmatrix}1\\1\\0\end{bmatrix} = \begin{bmatrix}0\\1\\0\end{bmatrix}$$

$$\begin{bmatrix}0\\1\\0\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix}\begin{bmatrix}a\\1\\2\end{bmatrix} = \underline{\alpha}, \underline{A_1}, \underline{A_2}, \underline{A_3} \equiv P_S = P_A + CTE$$

restando por renglon para comprobar

$$\begin{matrix}P_A\\CTE\end{matrix} \quad \begin{matrix}0100\\10\\\hline 0010\end{matrix} \quad \begin{matrix}0001\\10\\\hline 1111\\(negativo)\end{matrix} \quad \begin{matrix}0110\\10\\\hline 0100\end{matrix} \quad \begin{matrix}0010\\1111\\0100\end{matrix} \equiv \underline{A}, \underline{S}$$

DIFERENCIA ENTRE PROPIEDAD Y CONSTANTE

FIG. 4.2.1.E

3!.- Producto con una constante

$$M = P_n * CTE \equiv \underline{M} = f(\underline{P_a}, CTE)$$

Este algoritmo no fué desarrollado

4!.- División con una constante

$$D = P_n / CTE \equiv \underline{D} = f(\underline{P_a}, CTE)$$

Este algoritmo tampoco fué desarrollado

V.4.3. Operaciones incluyendo propiedades en R y en V.

Hasta el momento las soluciones de las operaciones sobre propiedades han sido o bien en estructura R o bien en V, y no se ha contemplado en el caso en que una operación incluya ambas estructuras.

Por ejemplo supóngase que en una consulta global, las propiedades que forman el predicado están unas en R y otras en V. Es claro que los métodos hasta aquí descritos no puedan aplicarse directamente, habrá que hacer una modificación.

Sin embargo, de manera general no siempre es posible descomponer una expresión lógica en dos partes, donde por un lado están las propiedades en V y por el otro propiedades en R. Para ser más claro daremos el siguiente ejemplo con una expresión lógica:

$(a+b) \cdot c$ donde a, b y c son propiedades
a y c en R
b en V

la idea sería encontrar

$$(a+b)\cdot c = b \; op \; f(a,c)$$

donde op. operando
f(a,c) función lógica

en este caso o primero se hace el

o lógicio (+) entre a y b

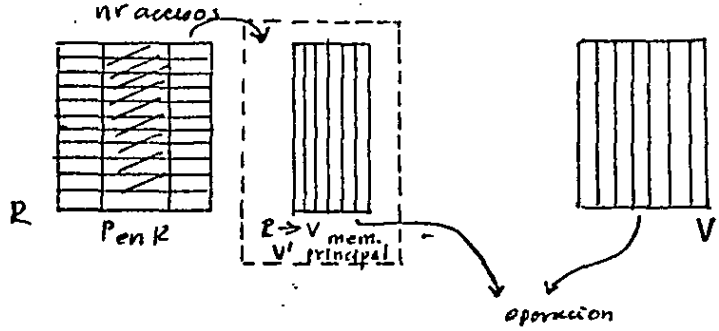
o se hace a • c + b • c, siempre

b tendrá que operarse o con a o con c

(propiedades en R) así en forma general una expresión no siempre puede dividirse en dos, para que se resolviera una parte en R y otra en V.

Por esta razón, la solución está en otro sentido.

Dado que se han desarrollado dos métodos o en R o en V. La solución que se plantea es las propiedades en R, se transforman en V (R→V) o viceversa (V→R). La transformación más adecuada se muestra con la siguiente figura.

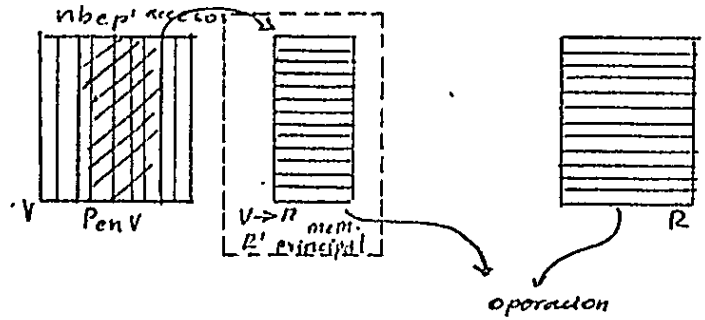


predicado - P en R y P' en V

VEC (vector de acceso)

TRANSFORMACION R→V

Fig. V.4.3.A



Predicado = P en R y P en V

nr= número de registros

nbep' número de bits en predicado en V

P en R - propiedades del predicado
que actúan en R.

P en V - propiedades del predicado que están en V

R' - archivo en R en memoria V' archivo V en memoria

R → V - transformación de R a V

V → R - transformación de V a R.

VEC (vector de acceso)

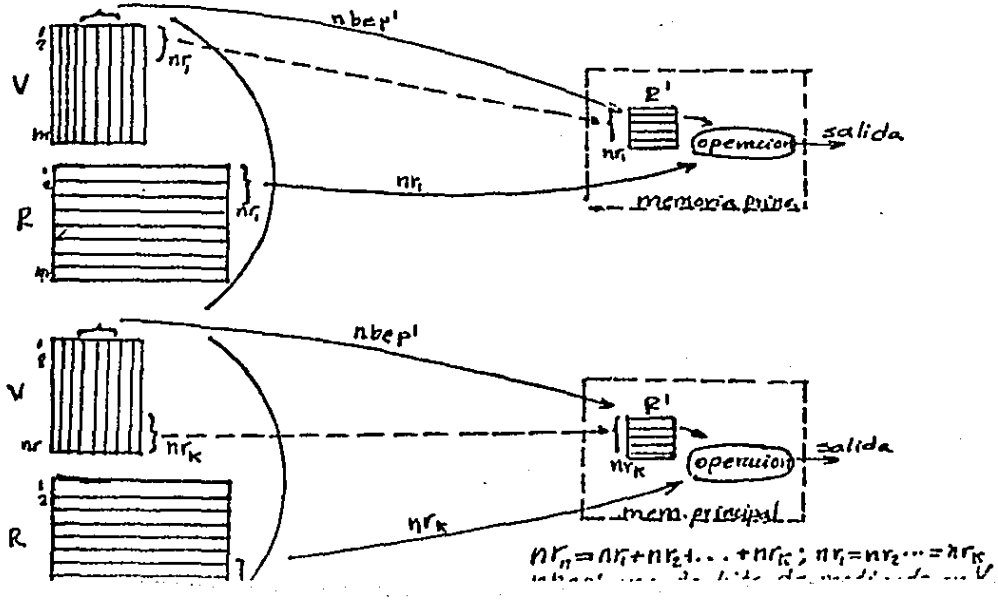
vector de acces
VEC

Se observa que la transformación R → V requiere de accesar todo el archivo para poder crear a V'. En el caso de V → R sólo se requieren accesar a los vectores involucrados, por lo tanto es más óptimo V → R puesto que sólo se accesa lo requerido.

Así, cuando un predicado incluye a propiedades en R y en V, el procedimiento a seguir es: se obtiene un archivo R' (V → R) en memoria principal y se resuelve como si en sí el predicado estuviese en R.

Las implicaciones que se tienen en esta solución, son: la utilización de memoria principal para la transformación V → R. Por lo tanto, se requerirá de memoria principal de trabajo dinámica (no se sabe la longitud del vector ni su número para dicha transformación, además se tiene que preveer que en el algoritmo de solución, si la memoria de trabajo disponible no es suficiente para resolver un predicado)

Con la siguiente figura se ilustra la solución cuando la memoria no es suficiente para la tansformación de R → V.



Sin embargo, dada que esta solución puede resultar costosa en memoria y en número de accesos, se puede incluir además el algoritmo que resuelve el caso cuando en una operación de consulta global se pueden separar en dos partes, una en R y otra V de tal forma que no se requiera transformación V → R. El algoritmo para este caso excepcional se puede resumir de la siguiente manera: toda expresión lógica se puede representar como una suma lógica ('+' -'O' lógico) de productos lógicos ('.'- "y" lógico) de pares de términos. Este algoritmo se aplica si existen dos grupos de pares, los que están en R y los que están en V. El algoritmo consiste: primero se aplica la parte en V, obteniéndo un vector de acceso intermedio, con el cual se accesará a R (para aplicar la segunda parte de la función) con los que no cumplieron (los que contienen '0's), los que cumplieron (los que tienen '1's) ya no vale la pena ver si además cumplen en la parte de R, basta con que se haya cumplido la función una sola vez (recordar que la función es una suma de productos), de esta manera los accesos en el peor de los accesos tendrá que ser igual a todo el archivo R (como en el caso de la otra variante), cuando de la parte V nadie cumplió y se requiere ver todos los de R por si ahí, alguno cumple, y en el mejor de todos los casos, cuando todos los registros cumplen en V, de tal forma

que ya no es necesario verificar la parte de R, estos son casos extremos, pero generalemente diremos que los accesos serán menor que n (n=no.de registros en R) por lo que será una solución más rápida que la primera variante y con la ventaja adicional de que no se requiere memoria principal, pues no se necesita transformación alguna. De esta manera el algoritmo se puede expresar:

dado un predicado F que pueda expresarse como:

$$\underline{f} = \underline{f}_v + \underline{f}_r$$

donde

f= función lógica del predicado

f_v= función lógica, expresada como una suma de productos de pares de propiedades en V.

f_r= igual que la anterior pero en R.

el vector solución será:

$$\underline{VEC} = \underline{VEC}'_v + \overline{\underline{VEC}'_v}.fr.$$

donde

$\underline{VEC}$- vector de acceso solución

$\underline{VEC}'_v$- vector de acceso intermedio aplicado a la función f_v

$\overline{\underline{VEC}'_v}$- negado del vector intermedio para aplicar a la función f_r

Así los accesos sobre R serán para aplicar fr serán:

na= VEC'_v

donde na en 'accesos' será:

nbp' $\leq$ na $\leq$ N

nbep' no. de bits en predicado en la parte solo de V

N= No. de registros en el archivo

Es claro que este algoritmo, de vector intermedio, en comparación con el de transformación VAR mejora la respuesta y memoria utilizada en los casos cuando f se puede expresar como fv + fr, sólo hay que tener en cuenta que este análisis siempre se tendrá que aplicar a todos los predicados, sin tener la seguridad de que se pueda aplicar el segundo algoritmo.

En la figura anterior se mostró, que cuando la memoria disponible no puede contener a todo el conjunto de vectores para la transformación V → R, R' se va formando por pedazos hasta haber recorrido a todo el archivo R, de esta forma el costo en 'accesos' de una operación con propiedades en R y en V es

knbep' + nr 'accesos'

donde k es el número de pedazos de V→R que puede contener en la memoria. En el capítulo de reconfiguración se volverá a hacer énfasis en el análisis de este costo y de todos en particular.

V.4.4. Operaciones entre archivos.

Como ya fué mencionado en el capítulo de estructura lógica, la operación entre archivos, se logra cuando las propiedades de un archivo, son identificadores de otros.

Así por ejemplo, en un archivo de artículos de una tienda donde tiene una propiedad=clave de proveedor, ésta es un identificador del archivo de proveedores, de tal forma que existe una relación entre articulos y proveedores, así que se puede realizar una operación incluyendo ambos archivos, por ejemplo cuales son los nombres y direcciones de los proveedores del estado de México, de cuyos artículos existen menos de cierta cantidad en bodega. Con la siguiente figura se ilustra la operación:

	CLAVE ARTICULO	NOMBRE	CANTIDAD EN EXISTENCIA	CLAVE PROVEEDOR
1	A1	ART 1	90	P_m
2	A2	ART 2	70	P_2
3	A3	ART 3	110	P_n
4	A4	ART 4	120	P_8
.				
n	An	ART_n	75	P_i

ARTICULOS

IDENTIFICADOR DE ARTICULOS
- CLAVE ARTICULO
- NOMBRE ARTICULO

	CLAVE PROVEEDOR	NOMBRE	DIRECCION	ESTADO	TEL.
1	P1	N1	D1	MEX	
2	P2	N2	D2	OAX	
3	P3	N3	D3	MOR	
m	P_m	N_m	D_m	MEX	

PROVEEDORES

IDENTIFICADOR DE PROVEEDORES
- CLAVE PROVEEDORES
- NOMBRE PROVEEDORES

RELACION ENTRE ARCHIVOS

FIG. V.4.4.A

Obsérvese que en el archivo de ARTICULOS la propiedad CLAVE DE PROVEEDOR es no identificador para ese artículo, pero si lo és para el archivo de proveedores, de tal forma que es esta propiedad no identificador- pero permite la relación entre archivos.

Ahora expresando la solución en términos de las funciones de acceso la pregunta se lleva a cabo como sigue:

PREDART= CANTIDAD EN EXISTENCIA < CTE

SLADART = 'sin salida'

PREDPRV = EDO=DE MEX

SALADPRV- nombre, dirección

CONSUL (ARTICULOS, PREDART, SALDART, Q, VSE, CLAVE PROVEEDOR)

CONSUL (PROVEEDORES, PREDPRV, SLAPRV, VSE, VS, CLAVE PROVEEDOR)

Con la primera función CONSUL, se obtiene del archivo de ARTICULOS (de todo, por eso vector de entrada =O) los que cumplen con el prédicado PREDART, es importante observar que el vector de salida (VSE) no va a apuntar o a proyectarse sobre el archivo de artículos, sino sobre el de proveedores (clave de proveedores es identificador de PROVEEDORES y no de artículos), de tal forma que ese vector de acceso de salida será el vector de acceso de entrada para la siguiente función CONSUL. También es importante hacer notar que en esta función no se requieren de ninguna salida del archivo de artículos.

En la segunda función partiendo de un subconjunto del archivo de PROVEEDORES dado por el vector de entrada VSE

(vector de salida de la función anterior) se le aplicará el predicado dado por SALPRV y se obtendrá un subconjunto que apunta a este mismo archivo (CLAVE DE PROVEEDORES, SI ES IDENTIFICADA DE PROVEEDORES) de los cuales la información pedida por la salida SALDPRV.

Con la siguiente figura desde el punto de vista de los vectores de acceso se ilustra la solución

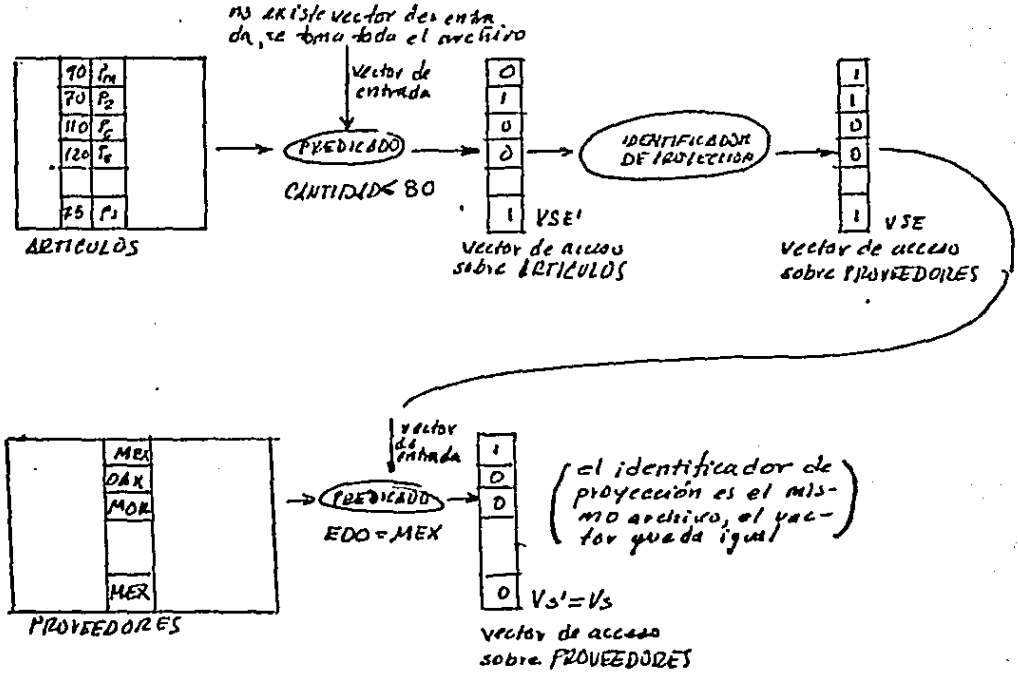


FIG. V.44.B.

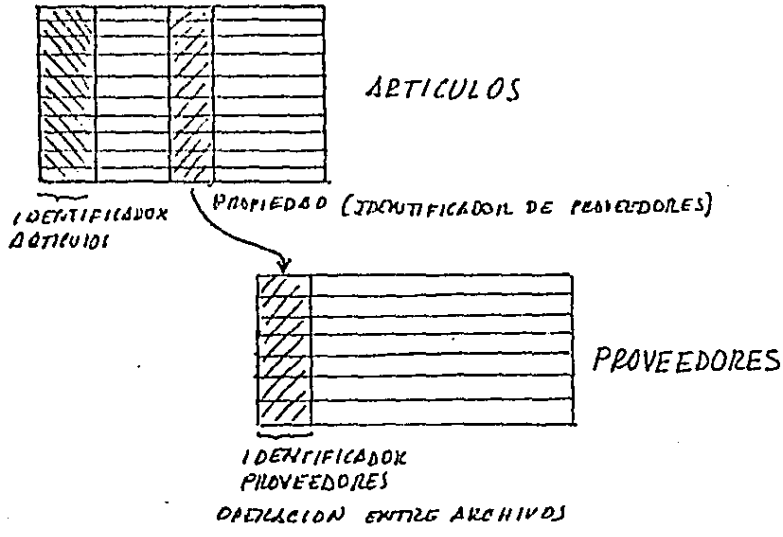


FIG. II.4.4.C

De esta forma, la solución a la pregunta propuesta en el principio, es dividida en dos partes, solución sobre el archivo ARTICULOS, proyectandola en PROVEEDORES (obtener del archivo de artículos, las claves de los proveedores donde la cantidad de existencia de artículos sea menos de cierta cantidad) y una solución sobre PROVEEDORES tomando sólo el subconjunto señalado en la primera operación (de las claves de proveedores antes obtenidas, toma del archivo de proveedores los que residen en el estado de México).

De la figura se observa que en la primera operación se obtienen un vector de acceso intermedio VSE' donde apunta al archivo de artículos, aquellos que la cantidad sea menor

de 80 entre los cuales están el 2 y el n (identificadores internos), dado que no se pregunta sobre artículos si no sobre proveedores, se toma la propiedad proveedores y se arma el vector de salida, así el artículo 2 apunta el proveedor 2 y el artículo n al proveedor 1, de tal forma que el vector de salida debe apuntar a estos proveedores. En la segunda función tomando de todos los proveedores solamente los señalads por el vector de salida de la función anterior se aplica el predicado EDO=MEX de los cuales solo cumple el proveedor 1. Es importante observar, dado que el vector de acceso se proyecta a sí mismo, el vector de acceso intermedio es igual al de salida. Así una vez obtenido el conjunto cumple con la condición, se obtiene de él la información especificada por la salida.

VI. RECONFIGURACION

La necesidad de reconfiguración en la base de datos, nació ante el hecho de dar al sistema generalidad y operación óptima sobre sus operaciones. En el capítulo de estructúra física, se muestra que para el tipo de operación a la que se orienta a la base, habrá implementaciones físicas mejores que otras, además, aunado este problema al de no poder saber con precisión, cuál es el comportamiento de un sistema en cuanto al flujo y operación de datos, la posibilidad de un sistema con una estructura óptima invariable (estática) no sería muy real. De esta manera, se hace necesario una configuración dinámica que se adapte al tipo de aplicación del sistema y a sus cambios de operación en el futuro.

VI.1 El problema en general.

La organización óptima, en cuanto a memoria utilizada y tiempo de respuesta, de una base de datos depende de la aplicación (tipo operaciones más comunes sobre la base) a la que se oriente al sistema. Esto significa que no existe una estructura única y óptima para un sistema general.

En el momento de creación de una base de datos, se puede conocer de alguna manera (implícitamente en la definición de la base o por información del usuario que la genera); el principal tipo de aplicación que tendrá la base, de tal forma que ésta podrá ser configurada con una estructura óptima de acuerdo a dicha aplicación.

Sin embargo, si la aplicación no es una característica estática del sistema, si no cambia a través del tiempo, así resulta que la optimización en el momento de creación, un tiempo después, al cambiar la base por los movimientos de datos que en ella se sucedan, la estructura ya no será óptima por no ser la más adecuada. Por ejemplo, propiedades que eran muy actualizadas son ahora muy consultadas, propiedades que eran muy poco comunes en registros, ahora lo son, etc.

Por lo tanto un sistema constantemente óptimo requiere de una reconfiguración contínua.

VI.2 Tipos de reconfiguración.

La reconfiguración la dividimos en dos tipos:

La que no modifica la estructura de la base, la información solo se reorganiza.

La que modifica la estructura de la base de datos.

En el primer tipo la aplicación no ha cambiado, sin embargo, la base de datos sí ha cambiado, debido a las actualizaciones de los datos, como son las altas, bajas y modificaciones de registros y/o propiedades de los archivos. De tal forma que la base para conservar un manejo óptimo tendrá que asimilar o actualizar cierta información. Este tipo de reconfiguración depende de la estructura y organización en particular de la base y las funciones a realizar serán tales como, integración de datos en áreas de desbordamiento a los archivos, reordenamientos de archivos, etc.

De esta forma en este tipo de reconfiguración los datos sólo se actualizan y se reorganizan sin modificar su estructura básica.

En el segundo tipo de reconfiguración, la aplicación (tipos de operaciones) ha cambiado en algunos archivos por lo que se requiere una restructuración que optimice la nueva aplicación.

Para este tipo de reconfiguración se requiere que al

menos la base, sea capaz de soportar dos tipos de estructuras diferentes. Por ejemplo para una organización de datos matricial se puede estructurar para dos aplicaciones, por columnas para consultas globales y por renglones para actualizaciones.

V.3 Método de reconfiguración, frecuencia de operación y costo de reconfiguración.

La reconfiguración consistirá en la evaluación de las diferentes estrusturas de la base y de la selección de la más adecuada. La evaluación permitirá conocer el comportamiento de la base a través del tiempo, lo cual reflejará el tipo de movimiento de los datos, por las diferentes operaciones que se realicen dada una cierta estructura; para posteriormente con un criterio de selección decidir o no el cambio a una estructura distinta en base a un óptimo aprovechamiento de tiempo (mínimo tiempo de respuesta) y memoria (mínima memoria utilizada).

Para poder evaluar una estructura se tendrá que medir el movimiento de los datos que en ella se suceden; de esta manera lo que define el movimiento de los datos es la frecuencia de operaciones que se realizan sobre registros y propiedades. Por ejemplo, definiéndo dos operaciones, consulta global y actualización, un tiempo después de estar usando la base habrá propiedades muy actualizadas, registros muy

consultados, etc., de esta forma se definirá la frecuencia de operación como el número de veces que se ha realizado cierto tipo de operación en un lapso de tiempo dado. Por ejemplo, una frecuencia de operación igual a 8 para la operación de consulta sobre la propiedad edad, significa que hasta ese momento la edad ha sido consultada 8 veces.

De esta manera en forma general, cada elemento del archivo renglón o columna, de acuerdo al tipo de estructura implementada deberá contener dicha información estadística, (frecuencia de operación).

	FCC	5	8	3	5
FRC	FCA / FRA	4	1	2	4
1	1	JUAN	25	M	3
2	5	MARIA	25	F	5
8	3	PEDRO	24	M	3
FRECUENCIAS DE OPERACION		nombre	edad	sexo	

frecuencias de operacion sobre propiedades

ARCHIVO LOGICO

FCC - frecuencia de consulta sobre columna
FCA - frecuencia de actualizacion -
FRC - frecuencia de consulta sobre renglon
FRA - frecuencia de actualizacion -

FRECUENCIAS DE OPERACION

FIG. V.3.1.A

En esta figura se ilustran las frecuencias de operación para una organización matricial que en el caso más general puede haber hasta cuatro tipos diferentes de frecuencias de operación para dos operaciones, consulta y actualización. Frecuencia de operación sobre registros de consulta y actualización (FRC y FRA) y frecuencia de operación sobre columnas de consulta y actualización (FRC, FCA). Por ejemplo, el registro María ha sido consultado dos veces y actualizado 5, hasta ese momento. La propiedad edad ha sido consultada 8 veces y actualizada dos.

El número de frecuencias utilizadas en una representación depende de las estructuras con que se cuente y del número de operaciones que se consideren.

El criterio de selección (costo de cambio). El criterio de selección consiste(dada una cierta estructura y el conocimiento del comportamiento de la base, a través de sus frecuencias de operación) en evaluar la estructura actual y las posibles, para así decidir por la [illegible] optima en memoria y tiempo de respuesta. De esta manera, el criterio consistirá en cuantificar en términos de memoria y tiempo de respuesta (costo de cambio), lo que le cuesta al sistema responder a las operaciones con la estructura actual así como cuanto le costaría con cada una de las otras posibles estructuras y

decidir por la del costo más bajo.

Cabe aclarar que se parte del supuesto de que el sistema tendrá un comportamiento, en un lapso de tiempo, igual al comportamiento que se midió en ese momento (a menos de usar una función predictora).

El costo de cambio tendrá que considerar dos costos elementales, el costo de operación y el costo de reconfiguración. El primero es lo que significa en memoria y tiempo de respuesta a las diferentes operaciones para las diferentes estructuras, y el segundo, es lo que cuesta en memoria y en tiempo, llevar a cabo físicamente la reconfiguración.

Por ejemplo, supóngase que una propiedad de un archivo por razones de operación justifique un cambio de estructura, pero sin embargo se dá el caso que el costo de reconfiguración física resulte muy alto, debido quizá a que el archivo es muy grande; así en términos totales no es aún justificable un cambio, pues la solución resulta más cara que el problema.

	RENGLONES	COLUMNAS	
ACTUALIZACIONES	20	30	COSTO DE OPERACION
	0	15	COSTO DE RECONFIGURACION
CONSULTAS	18	5	COSTO DE OPERACION
	0	10	COSTO DE RECONFIGURACION

COSTOS

FIG. III.3.2.

En esta figura se ilustran los costos en tiempo para las operaciones de consulta y actualización para una base de datos, con dos diferentes tipos de estructura (renglones y columnas), dada una estructura actual por renglones. En este ejemplo se ve que, aunque por consultas convendría una estructura por columnas, sin embargo, desde un punto de vista global la organización por renglones resulta aún adecuada. Deberá existir una tabla similar para los costos en memoria utilizada.

VI.4 Reconfiguración automática.

Los cambios de estructura de una base de datos puede ser realizada desde manualmente, por decisión del usuario o hasta totalmente automático y transparente al usuario. La primera es muy simple, pues no se requiere de un programa de decisión para reconfiguración; el trabajo lo realizará el usuario, lo que implica el conocimiento de las eficiencias de las estructuras utilizadas y del comportamiento de la base, lo cual generalmente será impráctico, a menos que el usuario sea muy experimentado y lleve él mismo las estadísticas de uso del sistema. El segundo método aunque un poco más complicado, evita al usuario preocuparse por estructuras.

Existe un punto intermedio el cual puede ser interesante y que puede ser opcional cuando se tiene un sistema automático, es aquel sistema que acepte sugerencias u órderes por

parte del usuario para cambios de estrucutra, es decir, un sistema semiautomático. Por ejemplo puede suceder que un sistema tendrá un cambio radical en cuanto a su aplicación, la cual se conoce, por lo que se podría forzar en un momento a la reconfiguración sin esperar que las frecuencias de operación lo reflejen un tiempo después. Esta información por parte del usuario podrá ser, no necesariamente decisiva pero sí importante para realizar una reconfiguración.

VI.5 Problemas de implementación en la reconfiguración.

El problema de la reconfiguración hasta aquí expresado, parte sobre el supuesto de una frecuencia de operación por propiedad independientemente de las demás frecuencias de operación de las otras propiedades. Es decir hasta ahora, las frecuencias de operación sobre propiedades sólo han contemplado consultas y actualizaciones sobre propiedades aisladas y no en conjunto.

Para ahondar más sobre el respecto, con el esquema hasta ahora presentado, para N propiedades se llevan 2*N frecuencias de operación (N para consultas y N para actualizaciones). Esto resultará adecuado si las operaciones son siempre sobre una propiedad, pero por ejemplo si no sólo se hace la pregunta a un archivo de empleados quienes tienen 30 hrs. de trabajo (lo que afectaría a la propiedad

horas de trabajo en su frecuencia de operación de consulta independientemente de las demás propiedades, y que de hecho lo és en este caso), se hiciera además el siguiente tipo de preguntas, incluyendo más propiedades, como quienes trabajan 30 hrs. a la semana y son mujeres. Con el mismo esquema a cada propiedad se le afectaría sus frecuencias de manera independiente entre ellas.

En este caso el esquema no es válido, porque no es lo mismo esa pregunta si en un caso las estructuras de la propiedad horas, está en renglones y la propiedad sexo está en vectores y en otro caso ambas propiedades están en vectores, puesto que si ambas propiedades están en vectores el número de accesos es igual al número de bits que componen ambas propiedades, pero si una propiedad está en R de nada sirve que la primera esté en V, pues ya no tomará el mismo número de accesos cuando ambas estructuras están en V; de esta manera las frecuencias de operación dependen en ciertos momentos de las demás frecuencias de operación, cuando hay una operación que involucra a varias propiedades.

Por esta razón para poder llevar un comportamiento total de cada una de las propiedades en un archivo habrá que considerar de manera general que las operaciones tendrían que considerar el número de posibles combinaciones que se

puedan hacer con todas las propiedades, es decir se requiere de un número de frecuencias igual a 2* (2EXP N-1) donde N es el número de propiedades en el archivo.

2 EXP N es el número de combinaciones que se pueden tener desde 0 hasta N términos, la posibilidad de 0 se excluye. Por ejemplo Si N=10 se requerirán de 2046 frecuencias de operación, que como se observa resultaría muy costoso llevar la estadística total.

De esta forma, dado el alto costo que tendría que pagarse por una estadística total sobre todas las posibilidades (no necesariamente usadas) en frecuencias de operación, la solución tendrá que tomar otro camino un tanto intuitivo para abatir dicho costo. Las soluciones a este caso son variadas. Una primera consideración a tomar será que las estructuras no serán excluyentes, es decir además de una estructura por renglones y de la de columnas se sumará la de renglones y columnas.

En términos generales ha sido planteado el problema de la reconfiguración; en el siguiente inciso se tratará el tema más concretamente sobre la estructura que se implementó.

VI.6.1 Reconfiguración afectando la estructura.

En la estructura de la base de datos implementada, como se vió ampliamente en el capítulo IV, se definieron dos diferentes forma de organizar las propiedades de un archivo, por renglones (archivo R) y por columnas (archivoV). Además como se mencionó en el inciso anterior, de una manera intuitiva, habrá propiedades contenidas en ambos archivos. Así cada propiedad puede estar representada de 3 formas:

R en el archivo R

V en el archivo V

R&V en archivos R y V

Las frecuencias de operación que se llevan para esta implementación serán sobre propiedades y sobre dos tipos de operación, consulta global y actualización. El problema de consulta sobre varias propiedades será tratado más adelante. Las frecuencias de operación se muestran en la figura siguiente:

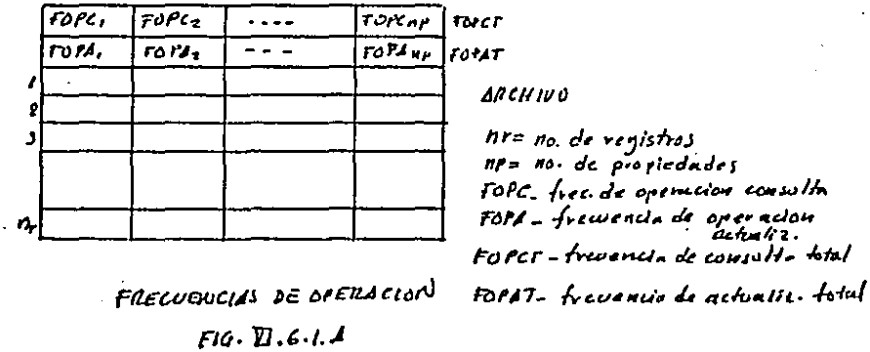


FIG. VI.6.1.A

También como se mencionó en las secciones anteriores la evaluación se hará sobre memoria utilizada y tiempo de respuesta, razón por lo cual se hará un análisis de tiempo y memoria para las dos operaciones, consulta global y actualización, y así finalmente tener los elementos para la decisión de la reconfiguración.

VI.6.1.1. Análisis de tiempo y memoria para consulta global.

El análisis de tiempo, se hace con el objeto de evaluar de una manera general los costos que en términos de tiempo de respuesta pueden producir las diferentes estructuras. Se puede decir que el factor tiempo de respuesta está en función de: la cantidad de tipos y operaciones que se realicen del número de registros y su longitud y finalmente de la forma en que físicamente se organicen los datos. Las frecuencias de operación, ya antes definidas, involucran los dos primeros términos (cantidad y tipo de operaciones) y los segundos, los agruparemos en un factor , que incluirá cantidad y forma de organización de los datos. De esta manera el tiempo de respuesta se definirá con la siguiente función:

$$T = f(FOP, I)$$

T- Tiempo de respuesta

FOP- Frecuencia de operación

I- Cantidad y organización de la información

De igual manera que para el tiempo, se puede evaluar

la memoria principal utilizada para llevar a cabo las operaciones. Haciendo una analogía con la función del tiempo, la memoria se define como:

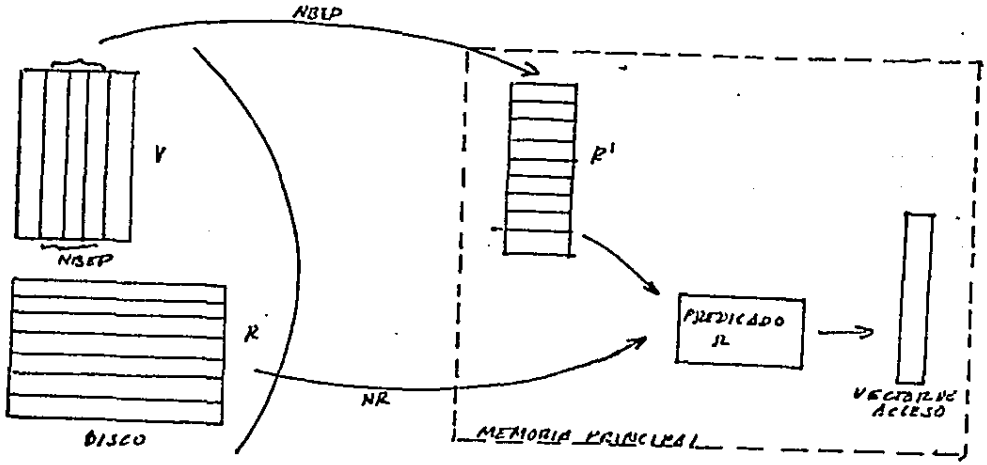
$$M = F(FOP, I)$$

Antes de analizar cada caso, repetiremos como se efectúa una operación en el caso cuando la consulta es sobre R&V.

La solución a este problema, justiticada en el capítulo de operaciones en la base de datos, consiste en pasar los vectores a un archivo de renglones R' residente en la memoria principal:

$$V \dashrightarrow R'$$

La solución puede ser ilustrada en la siguiente manera:



SOLUCION A OPERACIONES R y V

FIG. III.4.1.A.

También como se mencionó en el capítulo de operaciones, es posible, debido a la cantidad limitada de memoria principal con que generalmente se cuenta, la solución expresada en la figura anterior, será repetida varias veces (k), de tal forma que la parte copiada quepa en la memoria principal disponible. Así la operación se hará por partes hasta completar el archivo a operar.

Antes del análisis particular, se considerará la consulta global como una operación compuesta, la consulta en sí y la función de salida, puesto que esta última se verá también afectada por el tipo en particular de estructuras que tenga el archivo a operar, por ejemplo, después de una consulta global se puede pedir ddemás de la lista de quienes cumplen con el predicado la impresión de algunas de sus propiedades, las cuales podrán estar en R, V y/o R&V. Esta operación de salida tiene un efecto semejante al de la operación actualizaciones, puesto que se irá accesando registro por registro a través de sus identificadores, dados por el vector de acceso producido por la consulta. La diferencia de la operación de salida con la de actualización está en que la primera sólo requiere de un acceso, el de consulta, mientras que la segunda requiere de dos, el de lectura para consulta y el de escritura para la actualización en sí.

En la figura siguiente se muestra la operación consulta global considerando sus dos operaciones por separado, la consulta y la salida; para cada una de estas operaciones básicas tendrán la posibilidad de estar en los tres tipos diferentes de estructura, R, V y R&V, dando así, nueve tipos de solución diferentes para la operación consulta global:

TIPO	CONSULTA - SALIDA
1	R - R
2	R - V
3	R - RV
4	V - R
5	V - V
6	V - RV
7	RV - R
8	RV - V
9	RV - RV

TIPOS DE OPERACIONES GLOBALES

TABLA VI.6.1.1.B

A continuación también se describirán los términos empleados en el análisis:

nr_	no. de renglones
nbp_	no. de bits de propiedades por registros
nbpr_	no. de bits por registro físico en R
nbpsr_	no. de bits de propiedades solo en R
nbprv_	no. de bits de propiedades en R y V
nv_	no. de vectores en V
nsv_	no. de vectores de propiedades solo en V
nrv_	no. de vectores de propiedades en V y R
nbep_	no. de bits dentro de un predicado
nbep'_	no. de bits del predicado solo en V
nbei_	no. de bits dentro de una salida
nbei'_	no. de bits en la salida solo en V

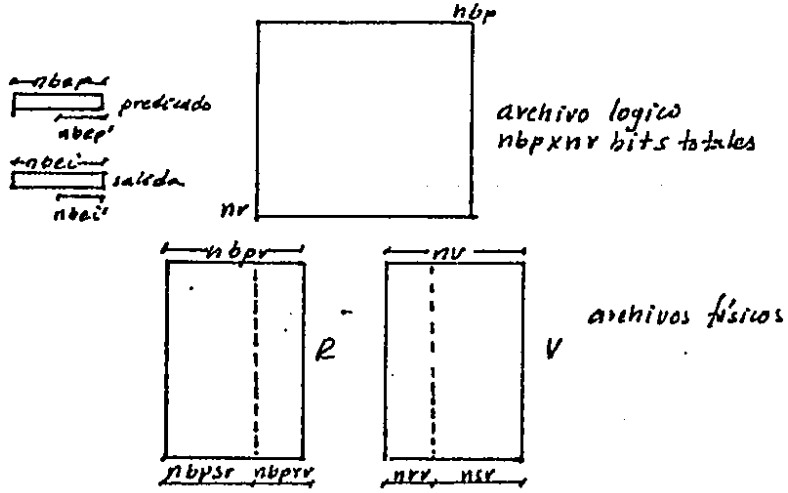


FIG. VI. 6.1.1.D

Para facilitar una comparación y obtener una idea cuantitativa del tiempo de acceso (proporcional al tiempo de respuesta) se definió la siguiente unidad: un tiempo de acceso igual a "acceso" el cual se obtiene al simplificar los términos cantidad de información y organización física, siguiendo la siguiente regla: accesar físicamente un registro o un vector, le toma el mismo tiempo expresado en la unidad "acceso". En la práctica esto puede funcionar más o menos cuando: los registros sean comparables en longitud con los vectores si R y V se bloquean de manera semejante (aunque realmente el tiempo de acceso varía de acuerdo al número de registros y de la manera en que se bloqueen y organicen los datos físicos).

Una vez definidas las unidades se describirán para cada de los 9 casos, sus dos términos principales, el que involucra tiempo de respuesta ("accesos" y bits accesados) y el que involucra memoria (bits en memoria principal). Solo hay que hacer notar que para la operación de salida, no se considerará el tiempo que se requiera en su generación, ya que dependerá del tipo de salida (papel impreso, archivo en disco, archivo en cinta, etc.), de esta forma, se considerará el tiempo de respuesta como el tiempo que le tome al sistema desde la búsqueda de la información para resolver el predicado hasta la obtención de la

última información requerida por la salida y no se considerará el tiempo de impresión.

Tipo RR

En este caso el predicado y la salida se resuelven al mismo tiempo; se accesa registro por registro, se ve si cumple con el predicado, si es así ya se tiene la información para la salida, sin necesidad de acceso adicional. De esta manera el total de bits que se requieren para realizar esta operación es de:

$$NBPR \times NR$$

y el numero de "accesos"

$$NR$$

para este caso, como puede observarse en la forma de solución no se requiere memoria adicional que la necesaria para contener un registro mientras es operado.

Tipo R-V

Este caso es similar al anterior en cuanto a la solución de la consulta, el problema que se presenta está en la salida, donde las propiedades requeridas están en V (una estructura no adecuada). Razón por la cual se hará necesario una transformación V → R', por lo que habrá que agregar a la expresión de consulta un nuevo término que involucra dicha transformación, este nuevo término puede ser expresado como

$$K * nbec * nr$$

y expresado en "accesos"

$$K\, nbec$$

En cuanto a memoria requerida, es necesario contar con cierta cantidad para poder realizar la transformación V→ R'; por lo que en cantidad de bit, se puede expresar como:

$$1/K\ nbec * nr$$

Tipo R-RV

En cuanto a la consulta, este caso es similar a los dos anteriores, y en cuanto a la salida es parecido al último, con el detalle que en este, la salida involucra a las dos estructuras, de tal forma que será necesario hacer una transformación V → R' (pero solo sobre las propiedades en V y son requeridas en la salida = NBEI') y así con R' y accesando R a través del vector de acceso tener la salida. De esta manera la expresión para la salida se compone de dos términos, el de la transformación V→R' y el de acceso por el vector, y puede ser expresado el primero en bits como:

$$K * nbei' \times nr$$

y expresado en 'accesos'

$$K * nbei'$$

y el segundo como:

$$nbpr * VEC$$

y en "accesos"

$$VEC$$

La expresión para memoria requerida es semejante al anterior caso, sólo teniendo en cuenta que solo una parte de los bits de salida están en V y son precisamente los que propiciaron la transformación V → R' (*nbec'*) así la memoria se expresa como:

$$1/K \; nbec' * nr$$

Hay que hacer notar que en cualquiera de los 9 casos si una propiedad está en RV, hay que considerar que se requiere doble espacio en disco igual a:

$$1/K \; nbec' * nr$$

Tipo V - R

Para este caso, la consulta cuenta con la estructura adecuada (para la solución del predicado, sólo se accesará las propiedades involucradas, no así en el caso R, que se accesa todo el archivo aunque no se requiera de la mayoría de la información). De esta manera, en cuanto a bits accesados la consulta se puede expresar como:

$$nbep * nr$$

y expresado en "accesos"

$$nbep$$

Una vez obtenido el vector de acceso a través del predicado, la adquisición de los registros que cumplen, se hará sobre los que se encuentren el vector (también no será necesario accesar información no requerida). Así este término se puede expresar en bits accesados como:

$$nbpr * VEC$$

y en "accesos"

$$VEC$$

En cuanto a memoria requerida no es necesario tener para alguna transformación.

Tipo V - V

En cuanto a la consulta se refiere, este caso es semejante al anterior, y en cuanto a la salida es similara a la de R-V.

De igual manera este término es similar al R-V

consulta : $nbep * nr$ bits
nr accesos

Salida : $K * nbei * nr$ bits
$K * nbei$ accesos

Tipo V-RV

El término para expresar la consulta es igual que para los dos casos anteriores. Y el término para expresar la salida es igual a la expresión de salida obtenida para R-RV.

La memoria requerida es igual que la de R-RV

consulta : $nbep * nr$ bits "accesos"
nr

salida: $K*nbei*nr$ bits
$K*nbei$ "accesos"

Tipo RV-R

La consulta en este caso, involucra dos términos, puesto que en el predicado están las dos propiedades, que de acuerdo a la solución que se le ha dado a este problema en el capítulo anterior (V.4.3) se vió que el número de accesos está entre $nbep'$ y nr . Así el primer término en bits accesados puede ser expresado como:

$$K * nbep' * nr$$

y en "accesos":

$$K * nbep'$$

El segundo término, en el cual ya se realiza la consulta en sí, incluye el acceso a R' (en memoria) y el de R para obtener el vector de acceso, así que en términos de bits accesados es:

$$nbpr * nr$$

y en "accesos"

$$nr$$

En cuanto a la salida, como en R-R, cuando se está resolviendo la consulta y se obtiene el elemento que cumple, como la salida está en R se tiene ahí la información para esta función, así no se requiere de accesos adicionales.

El término memoria, es claro que es diferente de 0 ya que se usa una transformación, de esta manera se expresa como:

$$1/K\ nbep'*nr$$

Tipo RV-V

La consulta en este caso es igual que la anterior.

La salida es igual que la de R-V.

La memoria requerida también igual a la de R-V.

Sólo hay que aclarar, el factor K para el primer término (k') es diferente para el de la salida (k''), pero en términos de memoria, se tomará el factor k más alto para formar su expresión, y no la suma, ya que estas operaciones consulta y salida, se realizan secuencialmente.

Tipo RV-RV

Este último caso es una combinación de dos, en cuanto a la consulta es igual que el anterior y en cuanto a la salida al de V-RV

tipo de predicado	Tiempo								memoria
	no. de bits accesados				"accesos"				no. de bits en memoria principal
	predicado		salida		predicado		salida		
	transformación	operación	transform.	operación	transf.	operación	transf.	operación	
R-R		nbpr x nr				nr			0
R-V		nbpr x nr +	K nbei x nr			nr +	K nbei		1/K nbei x nr
R-RV		nbpr x nr +	K nbei' x nr	nbpr x VEC		nr +	K nbei' +	VEC	1/K nbei' x nr
V-R		nbep x nr +		nbpr x VEC		nbep +		VEC	0
V-V		nbep x nr +	K nbei x nr			nbep +	K nbei		1/K nbei x nr
V-RV		nbep x nr +	K nbei' x nr	nbpr x VEC		nbep +	K nbei' +	VEC	1/K nbei' x nr
RV-R	K nbep' x nr +	nbpr x nr			K nbep' +	nr			1/K nbep' x nr
RV-V	K' nbep' x nr +	nbpr x nr +	K'' nbei x nr		K' nbep' +	nr	K'' nbei		* 1/K nbs
RV-RV	K' nbep' x nr +	nbpr x nr +	K'' nbei' x nr	nbpr x VEC	K' nbep' +	nr	K'' nbei' +	VEC	** 1/K nbs
	transformación V→R' para predicado	solucion predicado	transformación V→R' para salida	solución salida	transform. V→R' para predicado	solucion predicado	transform. V→R' para salida	solucion salida	memoria principal

* $1/K\,nbs = 1/K^{nbep'xnr}$ si $1/K^{nbep'xnr} > 1/K^{nbeix}$
en caso contrario
$1/K\,nbs = 1/K^{nbeixnr}$

** $1/K\,nbs = 1/K^{nbep'xnr}$ si $1/K^{nbep'xnr} > 1/K^{nbei}$
en caso contrario
$1/K\,nbs = 1/K^{nbei'xnr}$

ANALISIS DE TIEMPO Y MEMORIA PARA
LA OPERACION CONSULTA GLOBAL

Tabla VI.6.1.1.C.

Para dar una idea más clara y cuantitativa con respecto a tiempo y memoria principal que se requieren para las diferentes combinaciones de estructuras, daremos un ejemplo con datos ficticios, pero tratando de ser valores razonables.

Supóngase un archivo de 10,000 registros (nr=10,000) donde cada uno de ellos está formado por 225 bytes (nbp=1800) y físicamente están en dos archivos R y V (nbpr=1200, nv=800 algunas propiedades están en R y V) y se hace una consulta donde se involucran a 5 propiedades que en total representan 40 bits (nbep=40) es decir que en promedio, en estas propiedades, tienen 8 bits cada uno (28 posibles valores para cada una de ellas) de las cuales, en total de bits=20, están en V (nbep'=20) y el resto en R. La salida que se requieren para esta operación involucra a varias propiedades y en total son 75 bytes (nbei=600) de los cuales 300 bits de elbs están en V, y finalmente, 100 registros cumplen con el predicado (VEC= 100).

Se tabularán y graficarán bytes accesados, memoria principal requerida en bytes y en número de "accesos" (un acceso de registro-acceso de un vector). Se usará como punto de comparación, un archivo completamente en R (no en V), donde cualquier consulta global tendrá que accesar todo ese archivo.

A continuación resumimos los datos, se da una tabla con los vdoros obtenidos para diferentes factores de memoria (K=1,2...5) y finalmente gráficas comparativas

$nbp = 1600$
$nbpr = 1200$
$nr = 800$
$nr = 10000$
$nbep = 40$
$nbei = 600$
$VEC = 100$
$nbep' = 20$
$nbei' = 300$

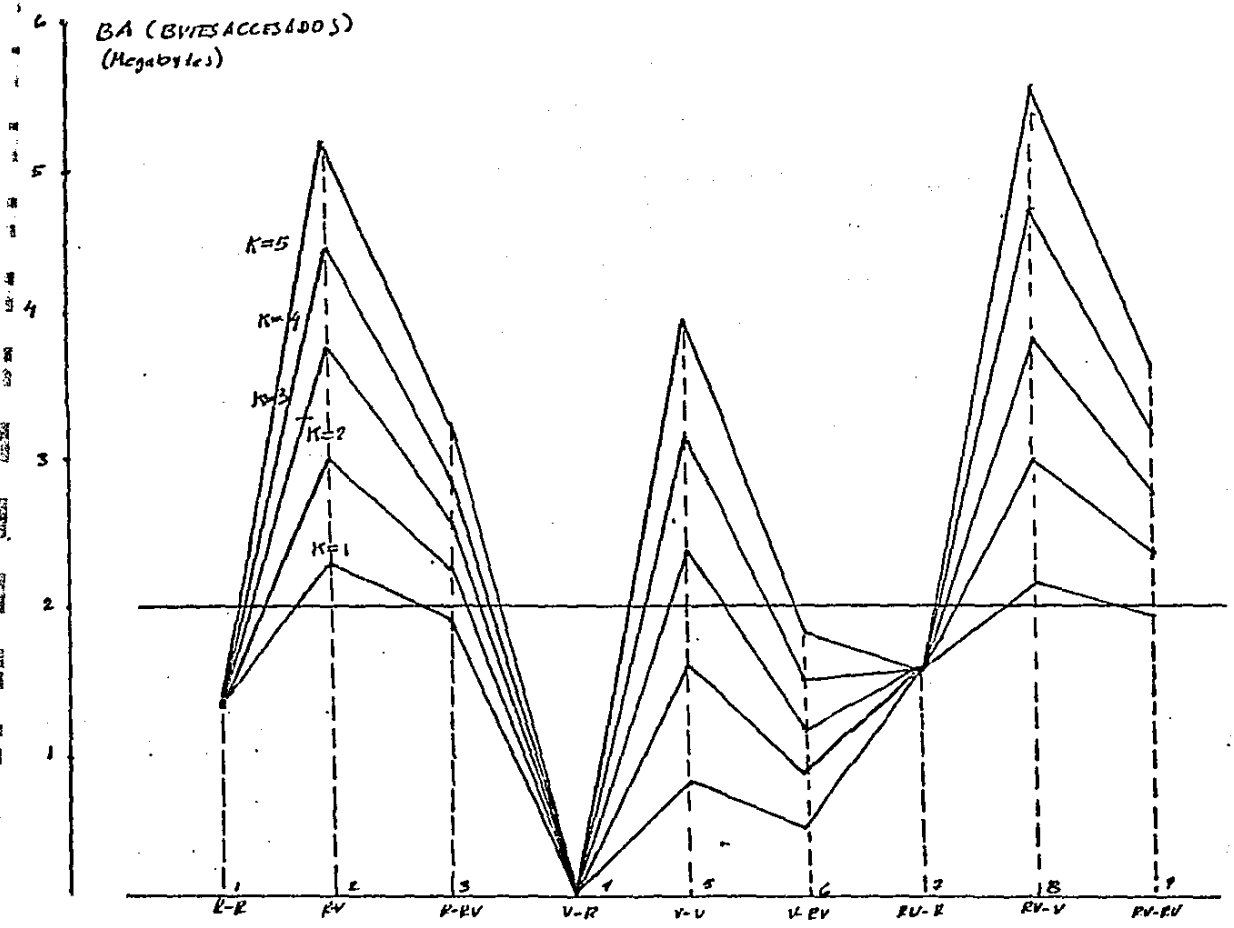
En la siguiente tabla se muestran para cada una de las diferentes posibilidades de estructuras y para varios valores de K (factor de memoria) los vabres que se obtienen en términos de total de bytes accesados (BA), bytes de memoria principal requerida (M) y el número de "accesos" requeridos ("A").

	K=→	1			2			3			4			5		
	TIPO	BA	M	'A'	BA	M	'A'	BA	M	'A'	BA	M	'A'	BA	M	'A'
1	R-R	1.5	0	10	1.5	0	0	1.5	0	10	1.5	0	10	1.5	0	10
2	R-V	2.25	750	10.1	3.	375	11.2	3.75	187.5	11.8	4.5	93.75	11.4	5.25	46.87	13
3	R-RV	1.89	375	10.4	2.26	187.5	10.7	2.63	93.75	11.0	3.0	46.87	11.3	3.375	23.43	11.6
4	V-R	.051	0	.14	.051	0	.14	.051	0	.14	.051	0	.14	.051	0	.14
5	V-V	.8	750	.69	1.55	375	1.24	2.25	187.5	1.84	3.05	93.75	2.44	3.8	46.87	3.44
6	V-RV	.9	375	.44	.375	187.5	.19	1.37	93.75	1.04	1.52	46.87	1.34	1.9	23.43	1.64
7	RV-R	1.52	25	10.02	1.55	12.5	10.04	1.57	6.75	10.06	16	3.375	10.08	1.62	1.68	10.1
8	RV-V	2.17	750	10.62	3.0	375	11.24	3.8	187.5	11.84	4.6	93.75	12.44	5.37	46.87	13.04
9	RV-RV	1.9	375	10.42	2.3	187.5	10.74	2.7	93.75	11.02	3.1	46.87	11.32	3.5	23.43	11.62

K- factor de memoria BA- bytes accesados del orden 10^6 (MB)
M- memoria principal requerida en bytes 10^3 (KB)
"A"- 'accesos' requeridos 10^3 (K)

FIG.III.6.1.1.E

Con los valores obtenidos y graficando BA contra todas los posibles combinaciones se obitne la siguiente gráfica.



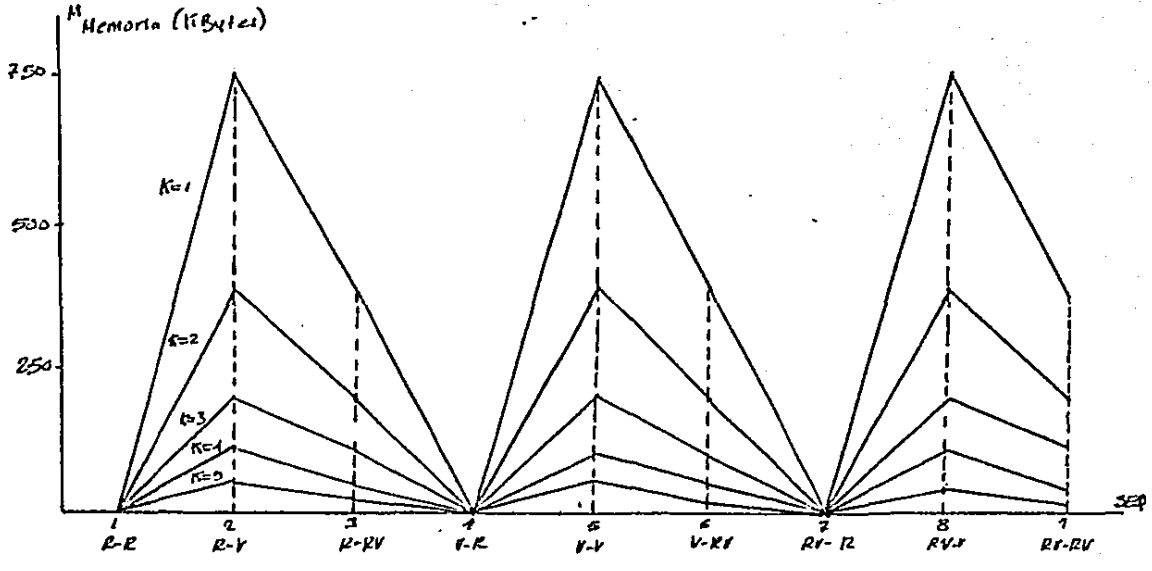
'ACCESOS' PARA LAS DIFERENTES ESTRUCTURAS

FIG. 6.4.F

En la gráfica obtenida la horizontal seq, es un medio de comparación y se obtiene pensando una estructura de archivo R secuencial, de tal manera que poder realizar cualquier operación de consulta tendrá que accesar el total del archivo R. Es claro que este archivo (R) será mayor que el archivo R con la estructura combinada, dado que algunas propiedades solo están en V por lo tanto Seq= nbpxnr= 2MB

De la gráfica se observa: que cuando se incrementa el factor k el tiempo de respuesta aumenta, excepto para R-R y V-R que son independientes de este factor. También se observará que la combinación de estructuras más adecuadas a la solución (en comparación con Seq) son R-R , V-R, V-RV, RV-R, siendo la óptima V-R, que es mucho menor que una estructura SEQ. Las estructuras con un valor muy alto R-V y RV-V lo cual se explica por qué en la consulta no se tiene la mejor estructura V y de igual manera para la salida, donde la mejor es la K.

Ahora tomando los valores de memoria para las diferentes estructuras obtenemos la siguiente gráfica.



MEMORIA PRINCIPAL PARA LAS DIFERENTES ESTRUCTURAS

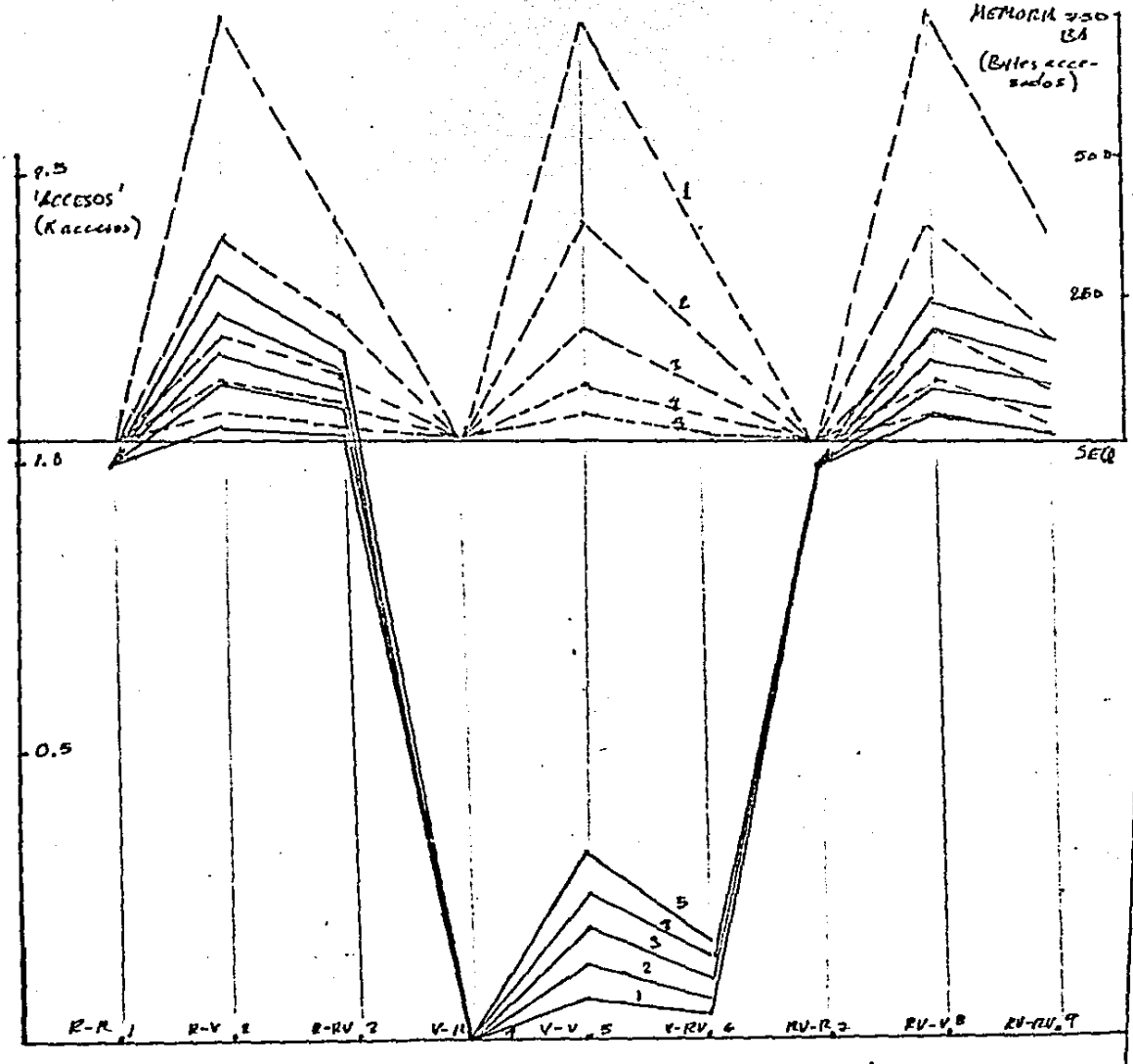
FIG-VI.6.1.1.E

En esta gráfica el punto de comparación es SEQ, en cuyo caso no requiere de memoria principal adicional por lo que su valor es cero.

De esta gráfica se observa, a diferencia de la de BA, que para un factor K muy alto, la memoria requerida es menor (en la de BA, mayor serán los bytes accesados que son proporcionales al tiempo de respuesta). También se observa que las estructuras más óptimas (iguales a SEQ=0) son R-R, V-R, RV-R y las de mayor valor R-V, V-V, RV-V.

Hay que señalar que el caso cuando la propiedad esté en RV es decir en ambas estructuras, con la ventaja de poderse usar en R o V según convenga al algoritmo, tiene un precio adicional, que el costo de la memoria en disco, el cual deberá de ser considerado en la evaluación en la reconfiguración.

Por útlimo, con los valores de "A", combinados con la de memoria se obtiene la siguiente gráfica:



COMPARACION ENTRE MEMORIA Y 'ACCESOS'

FIG. II.6.1.1-E

Con esta gráfica comparativa, se puede ver que la memoria utilizada es inversamente proporcional a los bytes accesados y por lo tanto al tiempo de respuesta, es decir a menos cantidad de memoria usada mejor tiempo de respuesta y a menor memoria usada peor tiempo de respuesta.

También se observa que las estructuras más óptimas considerando memoria y "accesos" son: R-R, V-R, V-RV siendo la mejor V-R. y las más costosas R-V RV-R.

Con los datos y gráficas obtenidas, las observaciones se pueden resumir como sigue:

Las gráficas aquí obtenidas no pueden generalizar cualquier operación, consulta global, puesto que depende de muchos factores, como es la cantidad total de información y la estructura de la operación, sin embargo a través de este ejemplo, particular se pueden obtener elementos del compartimiento de una base con las tres diferentes estructuras.

Primero la memoria se contrapone al tiempo, razón por lo cual habrá que hacer concesiones con estos dos factores, el medio de comparación para ambos, es el costo que en pesos cueste el tiempo y la memoria principal y secundaria de la máquina en particular que se utilice, de esta manera el costo decisivo es:

$$COSTOTAL = CSTM * M + CSTT * T$$

COSTOTAL - costo total
CSTM - costo de memoria (principal y secundaria)
CSTT - costo de tiempo
M - memoria T - tiempo

Sin embargo, es posible, que los términos de costo total se le pueda agregar un peso, para evaluaciones que involucran además de los costos otros factores, como por ejemplo rapidez de respuesta, aún sacrificando memoria de esta manera el costo decisivo puede ser:

$$COSTOTAL = CSTM * M * PESOM + CSTT * T * PESOT$$

PESOM- peso a memoria
PESOT- peso a tiempo

estos factores de peso aquí considerados dependerían de los casos en particular a tratar.

Otra observación interesante es que en una consulta global lo más óptimo es tener la parte de consulta en V y la de salida en R y lo peor es lo contrario.

De esta observación se deriva que las propiedades, para que puedan dar un buen tiempo de respuesta, se deben orientar para el tipo de operaciones a la que más frecuentemente sean usadas, objetivo que tratará de cumplir la reconfiguración.

VI.6.1.2 Análisis de tiempo y memoria para la actualización.

Para este tipo de operación, el análisis resulta ser muy sencillo, pues no es una operación compuesta. De esta forma solo habrá tres tipos de solución que corresponde a los tres

tipos de estructura, lo cual es mostrado en la siguiente tabla:

TIRO	No. BITS ACCS.	ACCESOS	MEMORIA REC.
1 R	2*NBPR	2	0
2 V	2*NBEP	2*NBEP	0
1 R&V	2*(NBPR+NPER)	2(1+NBEP)	0

tipos de operación con actualización

Tabla VI.6.2.A.

Para todos los casos no se requiere de memoria principal adicional para resolver esta operación. Además fundamentalmente la operación de actualización requiere de dos accesos para realizarse, el de lectura para la consulta y el de escritura para la actualización, de ahí el factor 2 que aparece en los términos de las expresiones.

A continuación se desglosan las expresiones para cada tipo de solución:

Tipo R.

Dado de que se cuenta con el identificador para llevar a cabo esta operación solo se accesa la información deseada, de tal forma que los bits accesados son:

2*NBPR

y el número de accesos: 2

Tipo V.

En este caso se requiere el acceso de todos los vectores involucrados en la actualización, de tal forma que en bits puede ser expresado como:

$$2*NPER*NR$$

y el número de accesos:

$$2*NBEP$$

Tipor R&V

En este caso será la suma de los anteriores, ya que se tendrá que actualizar a ambos archivos, en bits se puede expresar como:

$$2(NBPE+NBEP'*P)$$

y el número de accesos:

$$2 + 2*NBEP'$$

VI.6.1.3 Consideraciones para consultas que incluye varias propiedades.

Hasta quí, la forma de evaluación es válida para consultas sobre una propiedad. Para el caso en que intervienen más de una, la situación se complica y la solución como se mencionó en la sección problemas en la implementación de la reconfiguración, puede ser muy costosa por la gran cantidad de estadísticas que se tendrían que llevar (2*EXP NPS, NPS=No. de propiedades en el archivo).

Siendo costosa una estadística total se propuso la siguiente solución, llevando solo estadísticas por propiedad. Para ilustrar la solución emplearemos la siguiente tabla:

Supóngase que en un archivo 5 de sus propiedades y con las siguientes frecuencias de operación y estructuras.

Propiedad	estructura actual	FOPC	FOPA
P1	R	100	10
P2	R	30	700
P3	R	30	700
P4	R	30.	700

FOPC - frecuencia de consulta
FOPA. frecuencia de actualización

FRECUENCIAS DE OPERACION
TABLA VI.6-1.3.A

donde se puede observar que todas las propiedades excepto una (P1) están con la estructura adecuada, R para muchas actualizaciones, P1 estando en R se consulta más y por lo tanto, es "obvio" que deberá cambiar a V.

Pero supóngase que en lapso de tiempo entra la última reconfiguración y análisis de una nueva, hubo 200 preguntas de que incluirán varias propiedades de la siguiente forma:

P1 P2 - 30

P1 P3 - 30

P1 P4 - 30

P1 - 10

100 preguntas

y 300 actualizaciones sobre P2, P3, P4

y 10 actualizaciones sobre P1.

Al haber reconfigurado P1 y suponiendo con el comportamiento fué igual en el siguiente lapso de tiempo en ese análisis tendríamos:

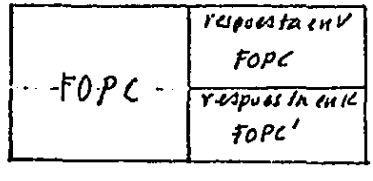
propiedad	extracción actual	Fopc	Fopa
P_1	V	200	20
P_2	R	60	1000
P_3	R	60	1000
P_4	R	60	1000
P_5	R	20	1000

TABLA VI.3 B

y de manera aparente es la estructura óptima, sin embargo, en la realidad no hay mejora en la solución, ya que antes la consulta se hacía sobre R para el tipo de predicados y ahora también se sigue haciendo sobre R (transformando en V en R') puesto que se consideró que se hicieron el mismo tipo de preguntas.

Para tratar de resolver esto, se planteó un factor correctivo (tal vez algo mas interesante sería uno preventivo) cuyo propósito sería avisar que tan bién resulto la reconfiguración.

De esta forma a la frecuencia de consulta se le agregó uno más



fopc- Es la frecuencia de operación que se ha llevado a cabo y que se ha resuelto en V.

fopc'- Es la consulta estando en V pero resulta en R.

Ahora volvamos al mismo problema, pero ahora incluyendo esta nueva frecuencia y sólo analizando las consultas.

I		II		III		IV	
Est	Fopc/Fopc'	Est	Fopc/Fopc'	Est	Fopc/Fopc'	Est	Fopc/Fopc'
P	0/100	V	45/90	V	50/150	V	00/180
R	0/30	P	0/60	V	30/0	V	60/0
R	0/30	R	0/60	P	0/90	V	30/0
R	0/30	R	0/60	R	0/90	P	0/120

Est-estructura actual

FRECUENCIAS DE OPERACION FOPC y FOPC'

Tabla VI.3.C

I Estado inicial

II Primera reconfiguración, con el campo solo se logró una mejora del 10% por lo tanto para mejorar la respuesta, se deberá cambiar otra propiedad, con el criterio del Fopc' mayor (en este ejemplo todos iguales).

III Para este caso ya hubo una mejora en P1 en un 33% (50/150), siguiendo el mismo criterio, pasamos al siguiente.

IV Ya en este caso P1 se mejoró en un 66% (120/180).

Es de observarse que si las propiedades siguen muy actualizadas deberán contener su estructura R, (es decir,

el cambio sería de R→RV) o en caso contrario de bajar las actualizaciones el cambio sería solo a V.

Para finalizar, habrá que señalar las limitaciones de este método.

Primero, se usa como un factor concetivo, es decir que primero se reconfigura y después se analiza, con más elemen tos si resultó adecuado. Quizá se tenga que buscar una solución inversa, un factor preventivo, en el cual se tengan ciertas reservas al cambio, cierta inercia, antes de lograr una reconfiguración más adecuada.

Segundo, el método para selección, qué propiedades habrá que cambiar para mejorar un cambio no bien logrado es un poco ciego (por falta de información), se relaciona al de mayor Fopc'. Tal vez se tengan que almacenar los patro nes de preguntas más usuales y a través de ellos mejorar los cambios sin mejora.

Para fines de este trabajo se utilizó este método.

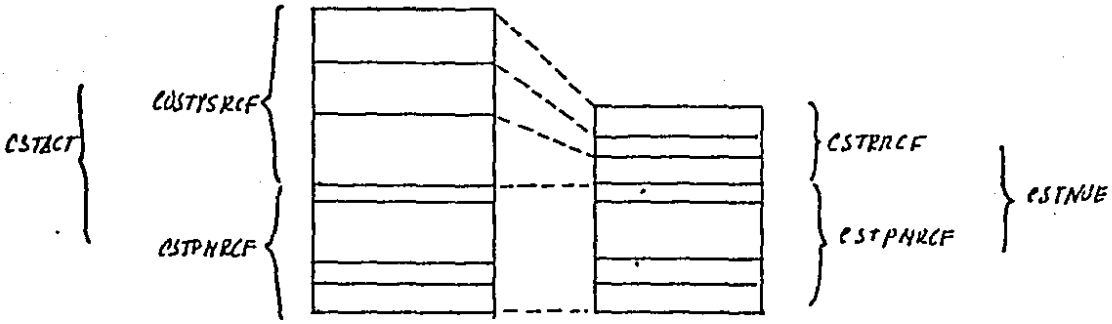
VI.6.1.4 Método de evaluación implementado.

La idea básica consiste en evaluar, en un lapso de tiempo, el costo de operación actual de un archivo, el costo con la estructura óptima de acuerdo a las operaciones realizadas hasta ese momento y el costo de cambio para pasar

de la estructura actual a la óptima.

Una vez evaluado estos tres costos, si el de la estructura óptima es menor o igual a la suma del costo actual y el de cambio, la reconfiguración se lleva a cabo, de lo contrario la estructura actual se conservará.

Con las siguientes figuras y expresiones definiremos el problema.



CSTACT — costo de la estructura actual
CSTPSRCF - costo de las propiedades que requieren reconfiguración
CSTPNRCF - costo de las propiedades que no requieren reconfiguración
CSTNUE — costo de la nueva estructura
CSTPRCF - costo de las propiedades reconfiguradas
CSTPNRCF - costo de las propiedades que no requieren reconfiguración

EVALUACION DE COSTOS
FIG. III.6.1.4.1.

$$CSTACT = CSTPNRCF + CSTPSRCF \quad \text{—— (1)}$$
$$CSTNUE = CSTPNRCF + CSTPRCF \quad \text{—— (2)}$$
$$CSTACT = CSTNUE + CSTCMB \quad \text{—— (3)}$$

usando (3) y sobstituyendo (1) y (2)

$$CSTCMB = CSTPSRCF - CSTPRCF$$

VI.6.1.4.1 Obtención de los costos actuales y con estructura Optima.

Estos costos (CSTPSRFC, CSTPRFC) se calcularán sobre cada una de las propiedades en su estructura actual, y de acuerdo a sus frecuencias de operación, la estructura óptima (posiblemente la actual). Así se obtendrá un costo total del comportamiento de un archivo. Es decir en la estructura actual de un archivo habrá propiedades, de manera general en R, V, y/o, R&V, y en la estructura óptima también de manera general habrá en R, V, y/o R&V, sólo que algunas propiedades cambiaron de tipo y algunas lo conservaron (por tener en teoría, una estructura adecuada). De esta manera estos costos se calculan como sigue:

$$CSTACT = \sum_{i=1}^{np} CSTPRO_i$$

$$CSTNUE = \sum_{i=1}^{np} CSTPRO_i$$

CSTPRO- Costo de la propiedad en el tipo de estructura en que se encuentra.

np= número de propiedades en el archivo.

COSTOPRO'= Costo de la propiedad con una estructura más óptima en base al comportamiento hasta ese momento de base.

donde:

$$CSTPRO = CSTM * MEM + CSTT * ACCESOS$$

CSTM — factor de costo de memoria (principal y secundaria) en costo/unidad de memoria

MEM — memoria utilizada

CSTT — factor de costo de procesador en unidades costo/tiempo

ACCS — accesos requeridos

VI.6.1.4.1.1. Calculo de los costos por propiedad.

Como se había mencionado antes, sólo se está llevando una estadística por propiedad, independientemente de las demás, lo cual puede traer como consecuencia, como también ya se mencionó, errores en el caso cuando las operaciones son sobre propiedades y no solo en una, razón por lo cual deberían llevarse estadísticas por propiedades y por cada una en particular, sin embargo, el costo es más alto para las estadísticas totales. De esta manera se propuso una solución (lo cual no significa ni la mejor ni la única). El problema donde más puede ser grave es en las consultas globales, como ya se hizo referencia anteriormente, y así tomando en cuenta estas consideraciones se incluyó un nuevo tipo de frecuencia de operación, consultas en R estando en V, que en otras

palabras influirá como un factor que corrige al factor de frecuencia, consulta; al aplicarse este factor de corrección implicará de manera implícita, las influencias que hayan tenido sobre el factor de consulta otras propiedades, en operaciónes que incluyen a varias.

De esta manera el costo por propiedad se calculará como sigue:

ESTRUCTURA	ACCS	MEM
R	ACCSR	MEMR
V	ACCSV	MEMV
R EV	ACCSRV	MEMRV+ MEMDSC

ACCS - ACCESOS
MEM- MEMORIA

TABLA DE COSTOS
TABLA VI.6.14.B

la estructura óptima, por propiedad es aquella cuyo

$$CSTPRO=CSTM*MEM+CSTT*\bar{T}.$$

es el menor. Puede que la estructura actual sea la óptima, por lo que en particular esa propiedad no requerirá de reconfiguración.

Ahora, se definirán cada uno de los términos en particular en función de las frecuencias de operación

1. ACCSR = FOPAC + FOPC * NR
2. ACCSV = FOPAC + NBPP + NC * NBPP + FOPC * NBPP
3. ACCSRV = FOPAC + FOPC * NBPP
4. MEMR = 0
5. MEMV = FOPAC * NBPP * NR
6. MEMRV = 0
7. MEMDSC = NBPP * NR (memoria en disco en RV).

Es de observarse que para la expresión 2 se usa el factor de corrección (como un método indirecto para tratar el problema de consultas con varias propiedades).

El hecho de aún estando en V y haber resuelto la consulta en R, aumenta el costo de consulta en V, a través del

factor FOPC', con este término se pretende la corrección, para que al evaluar y comparar las tres estructuras esta expresión tenga un peso más que debilite su posible selección.

Para el caso de memoria, en la expresión 4 y 5 (donde ambas tienen estructura R) vale 0, por que ambas operaciones no requieren de memoria auxiliar para su operación, pero no así con la estructura V en el caso de actualizaciones, por que se necesitará hacer la transformación V-R

Los términos costo de memoria (CSTM) y costo de tiempor (CSTT), pueden ser obtenidos por los costos en pesos que tengan estos dos recursos en la máquina en particular con que se trabaje.

De esta forma se obtienen, los costos por propiedad, los cuales se van acumulando, dando al final el costo con la estructura actual para un archivo, y por otro el costo con la estructura óptima.

Finalmente es importante hacer notar que las expresiones aquí usadas, son una forma muy simple del análisis previo que se hizo, y no cuenta con muchos factores de corrección, pero al menos en principio fueron especificados los factores que se tendrán que tomar en cuenta para un algoritmo de reconfiguración más completo y que en un futuro se puedan implementar.

VI.6.1.4.2. Cálculo de los costos de cambio.

Finalmente, una vez teniendo la forma de calcular los costos con la estructura actual y la óptima se obtendrá el costo de reconfiguración y así poder optar por la reconfiguración.

Este costo se medirá sobre los accesos requeridos para cambiar de estructura. Para esto definiremos los posibles cambios:

1 R x V

2 R x RV

3 V x R

4 V x RV

5 RV x R

6 RV x V

Estas diferentes transformaciones pueden realizarse por cuatro operaciones más elementales de reconfiguración que son;

1 CMBRXV

2 CMBVXR

3 BORRAR

4 BORRAV

la primera cambia de V a R, la segunda de R a V, la tercera borra una estructura R y la última borra a una V.

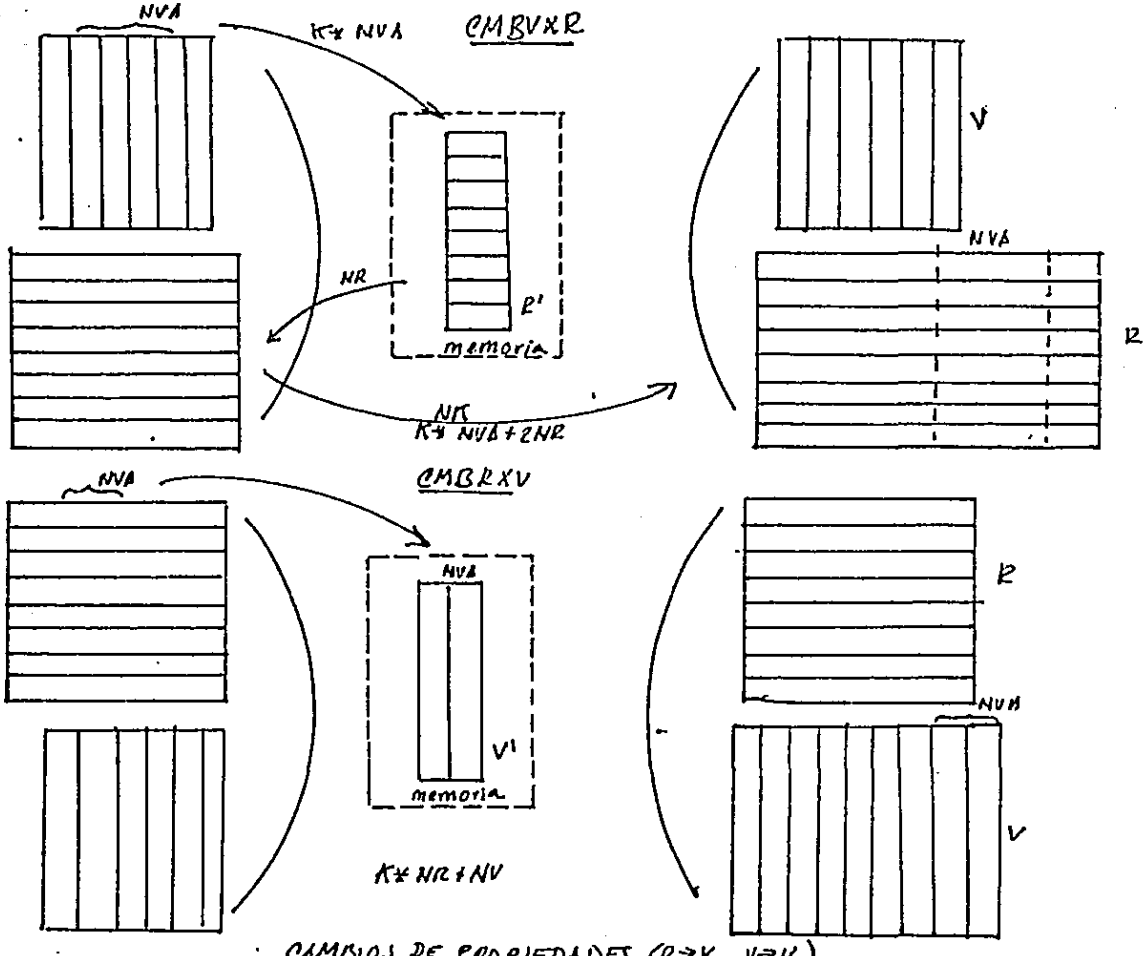
De esta manera se podrán calcular los costos de cambio de un archivo de la manera siguiente:

	CMBRXV	CMBVXR	BORRAR	BORRAV
RXV	K*NR+NVA		2*NR	
RXRV	K*NR+NVA			
VXR		NVA+2*NR		2*NVS-NVA
VXRV		NVA+2*NR		
RVXR				2NVS-NVA
RVXV			2*NR	

NR- no de registros en el archivo
NVA- no. total de vectores a actualizar
NVS- no. total de vectores en V

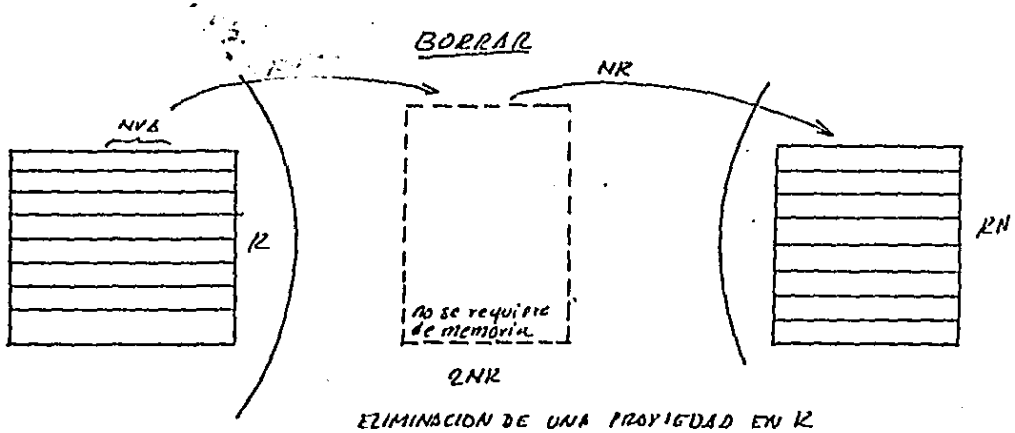
TOTAL DE COSTOS DE CAMBIO
TABLA VI.6.1.4.2.A.

huecos se lee primero todo el archivo V, (NVS) y posteriormente sólo se actualizarán del total solo los que requieran en el arvhivo (NVS-NVA). Para ilustrar las siguientes operaciones usaremos las siguientes gráficas:

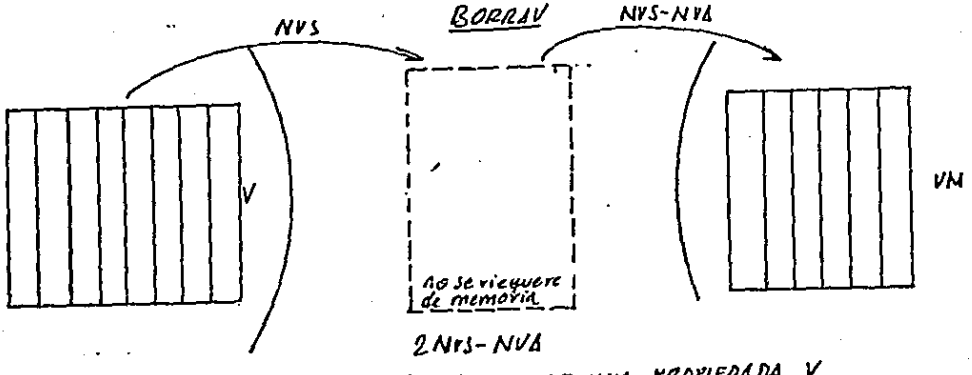


CAMBIOS DE PROPIEDADES (R→V, V→R)

FIG. III.6.1.4.1.1.B



ELIMINACION DE UNA PROPIEDAD EN R



ELIMINACION DE UNA PROPIEDADA V

FIG. II.6.1.4.1.1C

Actualización de V, y esto se debe a que al dar de alta nuevas propiedades en forma de vectores estas se incluyen al final de V, y no es necesario leer todo el archivo V.

BORRAR.- Esta operación requiere de 2 * NR accesos. Las primeras NR se hacen para leer el registro y el siguiente es para actualizarlo, una vez dado de baja, las propiedades no requeridas.

BORRAV.- Para esta operación se requiere de 2* NVS-NVA accesos. Para este caso a diferencia de CMBRXV, si se requiere leer de todo el archivo V, pues algunos de sus vectores, sufrirán baja de tal manera para que no queden huecos, se lee primero todo el archivo V (NVS) y posteriormente solo se actualizarán del total solo los que requieran en el archivo (NVS-NVA).

Para ilustrar las siguientes operaciones usaremos las siguientes gráficas:

Los algoritmos empleados en la reconfiguración están mostrados en las figuras anteriores, solamente hay que añadir, que una vez reconfigurado, se tienen que actualizar las tablas de la base, es decir algunas propiedades ya no tendrán la misma estructura, razón por lo cual habrá que cambiar apuntadores, actualizar valores etc.

De esta manera el costo de cambio se calculará, después de hacer el análisis del estado actual y posible reconfiguración, se verá que tipo de cambios se requieren y se obtienen el costo de cambio (CSTCMB).

Así una vez evaluados los costos se llevará a cambio la reconfiguración si:

$$CSTCMB \leq CSTPSRCF - CSPRCF$$

VI.6.1.4.3 Consideraciones sobre las limitaciones sobre el método de evaluación.

Es importante hacer notar dos aspectos, ya mencionados anteriormente, que pueden limitar el método de reconfiguración aquí descritos, estos son:

- la estadísticas de frecuencia de operación se lleva por propiedad. Aunque se plantea una solución por medio de un tercer factor de frecuencia.
- la reconfiguración parte de un comportamiento pasado (si el sistema no sigue el mismo comportamiento la

reconfiguración no puede tener efecto o aún más tener un efecto negativo.

El método aquí empleado, sobre todo en cuanto a cálculos y estadísticas pueden ser cambiados, no obstante el propósito principal de este capítulo fué ante todo, plantear el problema y los factores que intervienen (no se analizaron ni se buscaron algoritmos para obtener el más óptimo, tarea que puede ser llevada a cabo posteriormente).

De esta manera, además de las frecuencias de operación en propiedades de consulta y de actualización se agregaron para cada prioridad la consulta sobre R estando en V.

VI.6.2. Reconfiguración sin afectar la estructura.

Como se vió en el inciso "Tipos de reconfiguración", esta consiste en integrar, reordenar, etc., los datos a los archivos, sin un cambio fundamental en la estructura. Por razones de tiempo, este tipo de reconfiguración no se implementó, sin embargo, se mencionarán algunas consideraciones para una implementación futura de este tipo de reconfiguración para este sistema.-

VI.6.2.1. Areas de sobreflujo y de desbordamiento.

En un momento dado una actualización sobre un registro de archivo puede ser demasiado alto su costo (en una estructura V) quizá por otro lado la información no se

requerirá actualizada al instante. De esta forma es posible crear un subarchivo R (en disco) que contendría todos los nuevos registros, donde la consulta global se haría en dos partes, primero con el método antes mencionado y posteriormente una búsqueda secuencial sobre el archivo R' (de sobreflujo). Así el tiempo para integrar esta información dependerá como en todos los casos, del tiempo de respuesta requerido y de la cantidad de memoria disponible, para lo cual se requerirá del análisis de memoria y tiempo correspondientes. También se podría pensar en tener dos opciones de actualización alta, inmediata y no inmediata (para después de una evaluación del archivo R' integrarlos al archivo R).

El problema que se planteó en la estructura física en cuanto a propiedades cuyos valores generalmente no existen en el registro y se representan físicamente como propiedades explícitas, traen también algunos detalles que se tendrán que tomar en cuenta en la reconfiguración.

Físicamente para este conjunto de propiedades se les asignará un espacio fijo determinado. En el transcurso del tiempo al irse dando de alta esos valores, ese espacio se irá llenando, pero se puede dar el caso que en un momento dado, esa área sea sobrepasada y por lo tanto para no tener un

error, se hará necesario preveer areas de desbordamiento, de tal forma que en el caso de una consulta global o actualización, al ir al archivo R (estas propiedades siempre estarán en R) y al detectar una propiedad explícita y no encontrarla en su area correspondiente del registro en particular a tratar, habrá que ir a las áreas de desbordamiento para buscar el valor de la propiedad deseada.

Por esta razón al igual que las altas no inmediatas para las propiedades explícitas, se requiere de una evaluación de tiempo y memoria para poder realizar su reconfiguración.

VII Descripción del Sistema

Una vez descrita la estructura lógica, su estructura física y la forma de operar la base, en este capítulo se verá la forma de operar al sistema

Este capítulo se divide en dos partes, en la primera, se muestra el sistema total y en la segunda, con más detalle, se describirá la parte que se implementó del sistema total.

VII.1 Sistema General

Desde este punto de vista, el resultado final, será una base de datos orientados totalmente al manejo de datos e información, independiente de la estructura física de la base, donde la creación, operación reconfiguración, etc., serán operaciones que deberán realizarse independientemente del lenguaje de computación con que fué escrito y de la estructura física de la base y características físicas de la máquina en particular que se utilice.

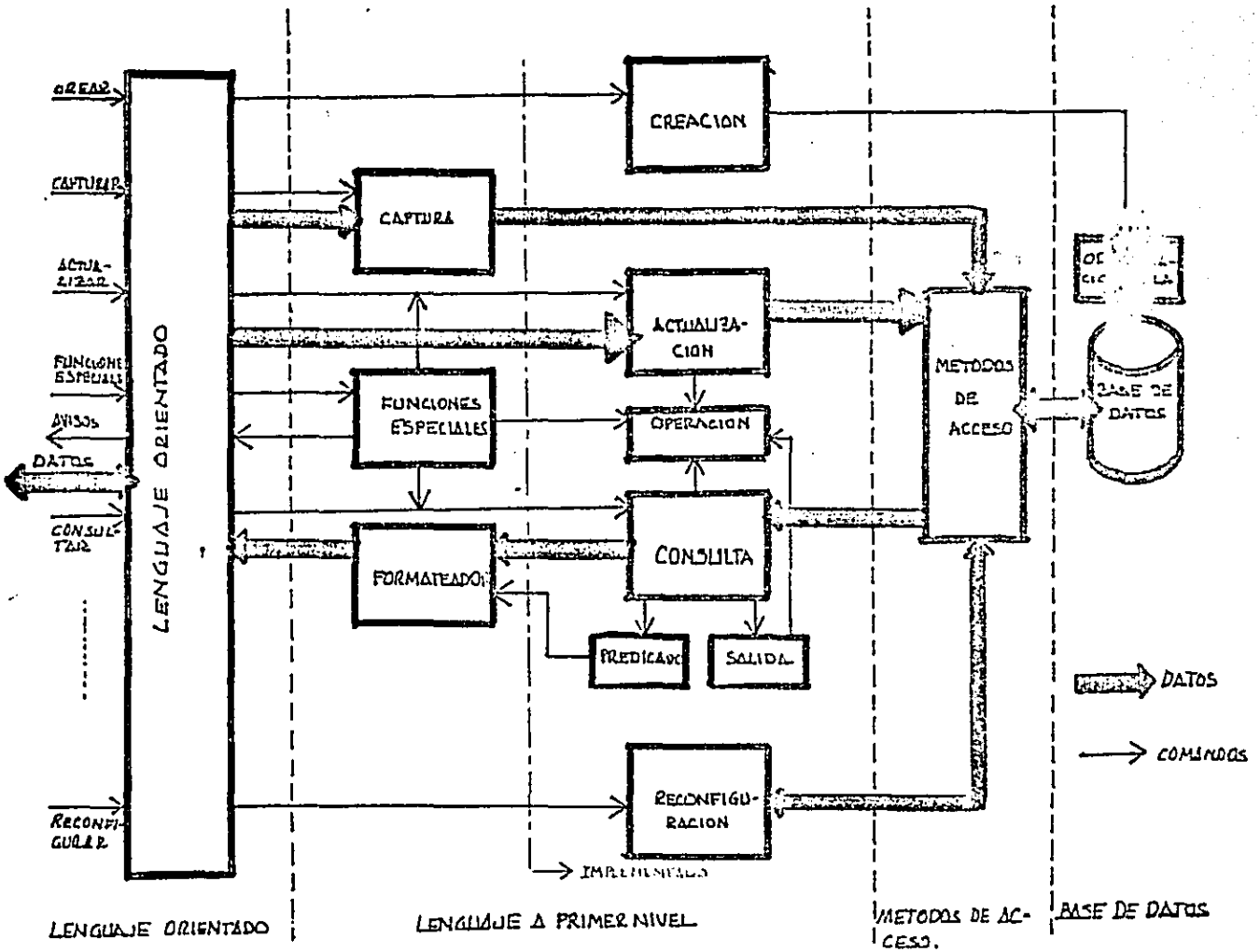
A continuación, en la siguiente figura, se describe el sistema general.

VII.1 Sistema General

Desde este punto de vista, el resultado final, será una base de datos orientados totalmente al manejo de datos e información, independiente de la estructura física de la base, donde la creación, operación reconfiguración, etc., serán operaciones que deberán realizarse independientemente del lenguaje de computación con que fué escrito y de la estructura física de la base y características físicas de la máquina. en particular que se utilice.

A continuación, en la siguiente figura, se describe el sistema general.

SISTEMA GENERAL DE INFORMACION RECONFIGURABLE



El sistema general está dividido en 4 secciones de la más básica hasta la más cercana al manejo de información, éstas son base de datos, métodos de acceso, lenguaje a primer nivel (comandos básicos de operación), lenguaje orientado.

VII.1.1. Base de datos

Esta área está muy cercana a las características de la máquina y a la estructura física de los datos. Aquí estarán contenidos todos los datos que conforman la base y la primer interfase entre datos físicos y lógicos que son las tablas que describen la organización lógica de la base y su relación física.

Esta área fué descrita en los capítulos de estructura lógica y física

Métodos de acceso

Esta área representa una segunda interfase entre datos físicos y lógicos que permiten obtener unidades de información (registros y propiedades) para ser operador.

Dentro de esta área están comprendidos todas las subrutinas necesarias para poder accesar la base en cualquiera de sus posibles estructuras y combinaciones.

Los métodos de acceso han sido descritos en los capítulos de estructura lógica, física y el de operación sobre la base.

Lenguaje a primer nivel (comandos básicos).

Dentro de esta área están comprendidas todas las subrutinas que permiten la operación de la base, a través de los métodos de acceso y de los algoritmos necesarios para realizar las operaciones. Estas subrutinas a través de comandos podrán ser utilizadas de tres maneras, como un primer lenguaje para manejo de la información independiente de la estructura, como intermedio para un lenguaje a más alto nivel, o como herramienta orientada a la información para la programación (usando llamadas a subrutinas) de aplicaciones especiales.

Los comandos básicos que se manejan en esta sección son los de creación, actualización (alta, baja modificación), consulta, operaciones y reconfiguración.

De esta manera estos comandos (llamadas a subrutinas) ya pueden ser utilizados para el manejo de información por un usuario o por un programa y además para ser utilizadas para programar operaciones especiales, como actualizaciones condicionadas por cambios en la información de la base, consultas automáticas por tiempo o por condiciones específicas, simulaciones, proyecciones, etc., que permitan analizar, proyectar, simular, etc., los datos de la base.

Lenguaje Orientado.

Esta área es el más alto nivel, en cuanto a la separación total entre lo físico y lo lógico, y comprenderá un lenguaje totalmente orientado a la información y al lenguaje de sistemas de información.

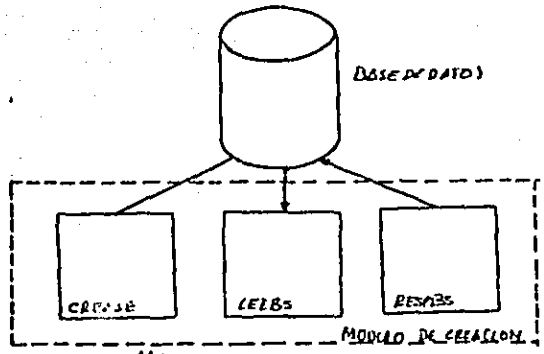
Esta área comprenderá un interprete que traducirá, de lenguaje orientado a comandos básicos, y ejecutará las operaciones especificadas.

En este nivel el usuario no tendrá necesidad de conocer características de la máquina ni estructuras físicas y su atención estará orientada al manejo y explotación de la información.

VII.2 Sistema inplementado

Dada la magnitud del sistema general antes descrito, la implementación, para propósitos de la tésis, sólo contempla la base de datos, métodos de acceso y lenguaje a primer nivel. No obstante no se dejó de preveer estas tres áreas dentro del sistema general de tal manera que el último punto, lenguaje orientado, puede desarrollarse, en principio sin el menor problema y sin afectar la estructura básica, de igual manera para las aplicaciones especiales usando el lenguaje de primer nivel.

La base de datos fué ya descrita en dos capítulos de



MODULO DE CREACION

FIG. VII. 2. 1. 1. A

CRBASE (NB) Esta subrutina se ejecuta al momento de la creación de una base y su objetivo es definir la estructura física inicial.

NB Es un identificador de la base.

La elección, de la estructura inicial se hace en base al tipo de propiedad (implementación implícita, como se vió en el capítulo de estructura física), es decir propiedades con valores indefinidos ya sean numéricos o alfanuméricos se representan en R y a la inversa propiedades con valor definidos se representan en V.

estructura lógica y física, métodos de acceso también los capítulos mencionados y además en el de operación de la base. Por lo tanto, sólo resta explicar el lenguaje a primer nivel y su modo de operarlo.

VII.2.1 Lenguaje a primer nivel (comandos básicos)

Este inciso se dividirá en tres partes, una descripción global de cada uno de los comandos, las funciones elementales y la forma de utilizarlos.

VII.2.1.1. Diagramas de bloques de los comandos básicos

Los comandos básicos del sistema, son un conjunto de subrutinas que permitirán realizar las funciones más básicas de este sistema de información, y estos son creación, actualización, consulta y reconfiguración. Todos estos comandos, excepto el de creación, requieren de los métodos de acceso para la obtención de los datos de la base.

VII.2.1.1.1. Módulo de creación

Este subsistema tendrá la función de crear la organización de una base y de incrementarla. El comando de creación solo creará el esqueleto de la estructura de la base (los datos en sí, serán almacenados por el subsistema de captura).

En el siguiente diagrama de bloques se describe este módulo:

Esta subrutina se manda llamar cuando se requiere crear una base.

LEEBS (NB). Esta subrutina obtiene la estructura de la base (NB) de disco a memoria. Esta subrutina se emplea siempre al principio cuando se va a operar la base. El sistema implementado automáticamente llama a esta subrutina. En un programa de aplicación especial es la primera llamada que se debe hacer.

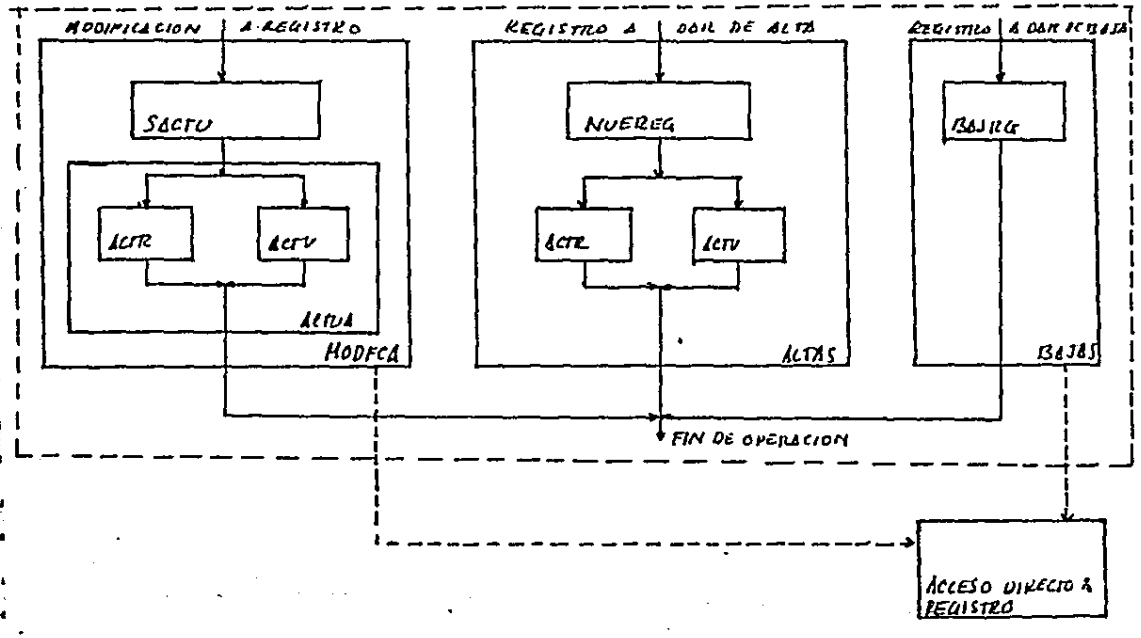
RESPBS(NB). Esta subrutina es inversa a la anterior y su objetivo es proteger en disco la estructura de la base, que después de haberle realizado operaciones las estadísticas sobre las propiedades han cambiado. En el sistema implementado al finalizar con la base automáticamente llama a esta subrutina, también al llamarse CRVASE esta al final automáticamente llama a esta subrutina. En un programa de aplicación es la última llamada que debe hacerse.

Dentro de este módulo se puede incluir (en el sistema

no se implementó) una subrutina más, MODFBS, cuyo objetivo sería cambiar algunas características de la estructura, tales como crear nuevos archivos, dar de baja a algunos, modificar archivos a través de sus propiedades, creando nuevas, quitando algunas o cambiandolas (aumentando campos, reduciendoles, etc.). La realización de este procedimiento haría uso de algunas de las subrutinas de reconfiguración.

VII.2.1.1.2 Módulo de actualización

La función principal de este subsistema, es las operaciones de actualización que se realizarán en la base, tales como modificaciones, altas y bajas.



MODULO DE ACTUALIZACION

FIG. VII. 2.1.1.2.A

Este módulo está compuesto por tres procedimientos importantes MODFCA, ALTAS, BAJAS

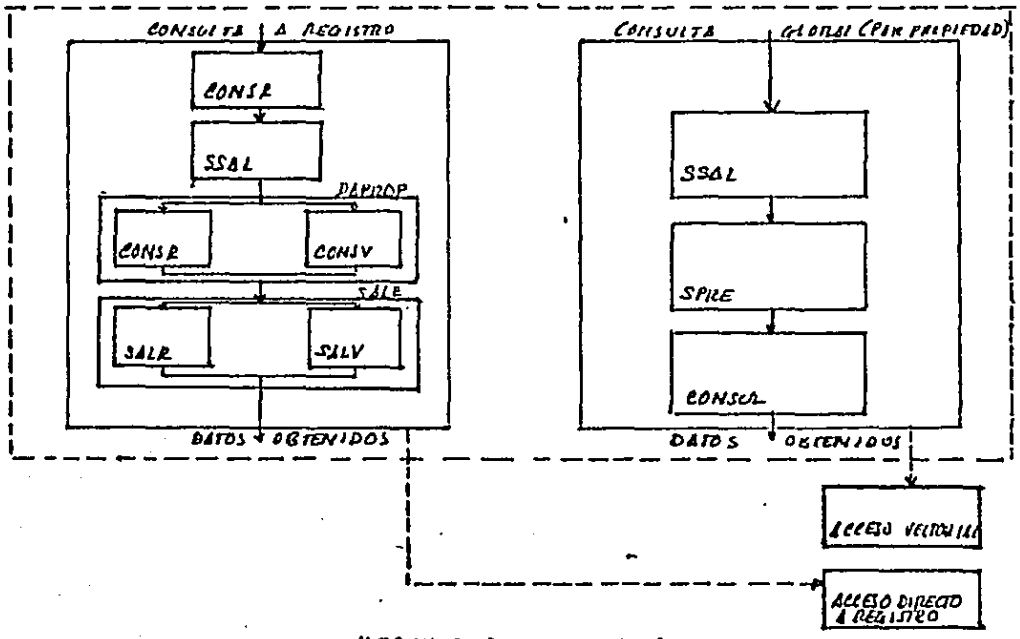
MODFCA, Este procedimiento se encargará de hacer cambios en los valores de algunas de las propiedades de un registro en particular, por lo tanto, utilizará la función de acceso directo a registros. Este procedimiento a su vez, está dividido en dos SACTU Y MODFCA.

SACTU - Se encarga de crear una tabla que contenga todas las propiedades y nuevos valores a modificar, una vez checados los nombres y los valores. Una vez obtenida esta tabla se actualizará en R (ACTR) o en V (ACTV) según sea el caso.

ALTAS, El objetivo de este procedimiento es incluir un nuevo registro dentro de un archivo. Estará formado principalmente por tres procedimientos, NUEREG (que se encargará de checar los nombres de propiedad y valores, así como de incluir en las tablas del sistema) ACTR y ACTV (que son las mismas funciones para actualizar en R o en V).

VII.2.1.1.3. Módulo de consulta

El objetivo principal de este subsistema será la consulta de la base de datos, para lo cual habrá dos procedimientos importantes RCONS SCONS, donde el primero se encargará de la consulta a un registro y el segundo de la consulta global. En la siguiente figura se ilustra con un diagrama de lo que es el módulo de consulta.



MODULO DE CONSULTO

Fig. VII.2.1.1.3.A.

RCONS.- Este procedimiento se encargará de la consulta de registros en particular, por lo que usará el acceso directo a registros.
Este procedimiento está formado a su vez por tres principales, CONSR, SSAL, DAPROP.

CONSR.- Se encargará de checar la información pedida y de crear una tabla con las propiedades requeridas.

SSAL.- Se encarga de generar el formato de salida que se requiere.

CONS.- Se encarga de hacer el acceso, a través de DAPROP (que a su vez lo hace o en R (CONSR) o en (CONSV) y de generar la salida (SALE) especificada por SSAL (que puede ser en R (SALR) o en V (SALV).

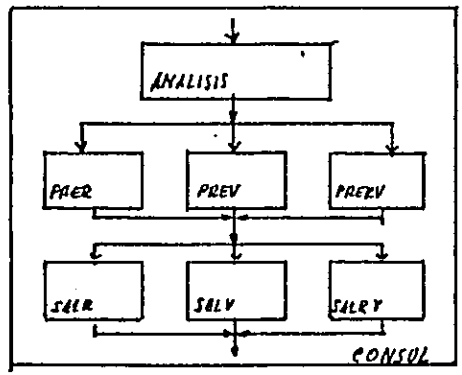
SALE.- Procedimiento que despliega la información requerida, la cual puede estar en (SALR) o en V (SALV)

SCONS.- Este procedimiento se encarga de la consulta global y tiene 3 procedimientos importantes SSAL, SPRE, CONSUL.

SSAL.- Este se encarga del formato de salida que se requiere.

SPRE.- Este procedimiento se encarga de la creación y validación de los predicados (condiciones al archivo).

CONSUL.- Este procedimiento es el que se encarga de la consulta global. Este procedimiento está formado por otras más simples, las cuales se mostrarán en la siguiente figura



MODULO DE CONSULTA

FIG. XII. 2.1.1.2.B

ANALIS.- Este procedimiento se encargará de analizar el predicado y la salida, para proveer que tipo de solución se le vá a dar (en predicado PRER, PREV, o PRERV, en salida SALR, SALV o SALRV) de acuerdo a la estructura (R, V, RV) en que se encuentren las propiedades en cuestión. Además se encargará de llevar las frecuencias de operación.

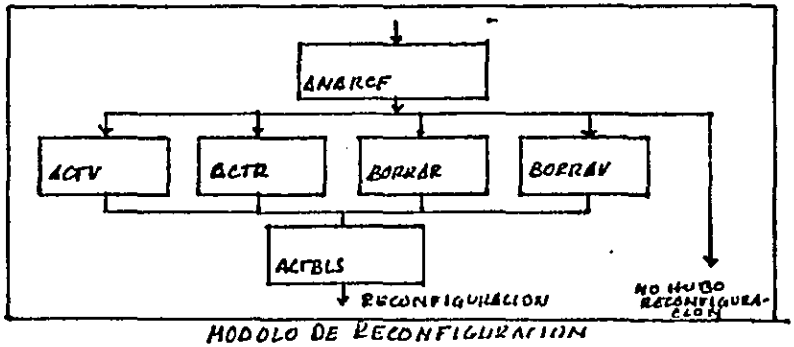
PRER, PREV, PRERV.- Procedimientos que resolverán los predicados de acuerdo a la estructura implicada (R, V, RV)

SALR, SALV, SALRV.- Procedimientos que resuelven la salida (especificada en SSAL) de acuerdo a la estructura implicada.

VII.2.1.1.4 Módulo de reconfiguración.

La función de este subsistema es evaluar la reconfiguración y llevarla a cabo si ásta se justifica.

Este módulo consta de las siguientes funciones básicas:



MODULO DE RECONFIGURACION

ANARCF.- Este procedimiento se encarga de la evaluación del costo sistema actual con las frecuencias de operación llevadas hasta ese momento, así como la evaluación del costo con la estructura óptima y del costo de cambio. Para así finalmente llevar a cabo o no la reconfiguración. Además determinará las funciones de reconfiguración que se tendrán que llevar a cabo si se decide por la reconfiguración

ACTV.- Este procedimiento es una función básica de reconfiguración y su objetivo es pasar propiedades de R a V.

ACTR.- Procedimiento para pasar propiedades de V a R.

BORRAR.- Procedimiento para borrar una propiedad R

BORRAV.- Procedimiento para borrar una propiedad V.

El objetivo de este procedimiento es restaura las tablas que describen al sistema al término de una reconfiguración.

VII.2.1.2. Funciones elementales y de operación.

Este tipo de funciones están relacionadas con el acceso y con cierto tipo de operaciones que facilitarán el manejo de los datos. A continuación se describen estas funciones:

VAL=DAPROP (ARCH, PIDENTI=VAL, PROP, E).

Esta función obtiene el valor de la propiedad PROP del registro cuyo identificador PIDENTI es VAL, del archivo

ARCH. El resultado estará dado en VAL. Si hubo algún error estará dado por E.

MTPROP (ARCH, PIDENTI=VAL, PROP, VALP, E)

Esta función es la inversa de la anterior, el valor de propiedad VALP es actualizado.

DIR=DADIRV(VEC)

Esta función obtiene de un vector de acceso VEC de manera secuencial, el siguiente registro que apunta (DIR), si ya no existe más DIR=O

MT DIRV(VEC,DIR)

Esta función es inversa, se hace que el vector apunte al registro, indicado por DIR.

OPLOVE (OP,VR, VI, V2)

Esta función permite realizar operaciones lógicas entre vectores VR ← VI-OP-V2 donde OP son las funciones lógicas: negación (-0 lógico (2), y lógico (3), o exclusivo (4).

En el inciso VII.2.1.3.2. se verá un ejemplo de como podrían aplicarse otras funciones.

Los comandos de captura y formateador no fueron implementados al sistema por falta de tiempo y por tener más interés en desarrollar la parte más operativa que la de entrada y salida.

El desarrollo de los programas de estos comandos no es complicado pero si requiere tiempo para implementarlos, y el no incluirlos en este sistema no cambia en nada la filosofía del mismo.

VII.2.1.3. Uso de los comandos.

En el lenguaje a primer nivel existen dos posibilidades para la operación de la base, usando los comandos básicos simplemente o usandolo junto con funciones más básicas o incluso auxiliándose con un lenguaje de programación, cuando la aplicación específica lo requiera.

VII.2.1.3.1 Comandos básicos.

Para una operación más o menos típica (creación consulta individual o global, actualizaciones, bajas, altas y modificaciones capturador y formateador) son suficientes los comandos básicos para poder operar la base. A continuación se describen cada uno de ellos usando ya el sistema:

Ejemplo de CREAR: (Comando para crear una base)

Base de datos de 1 archivo- EMPL

y 6 propiedades.

Propiedades

CLVE - Clave del empleado 10 dígitos

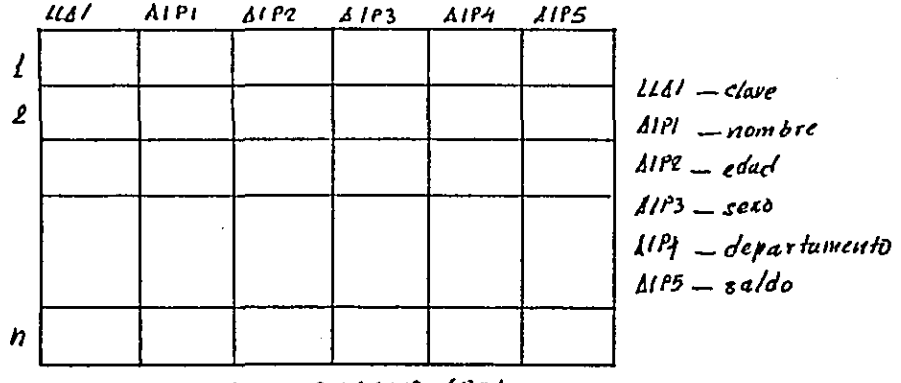
NMBR - Nombre del empleado 6 caracteres

EDAD- de 18 a 60

SEXO - Masc o fem

DPTO - departamento (A o B o C o D)

SLDO- hasta 5 dígitos en pesos (no hay categorías)



ARCHIVO LOGICO A1201

FIG. VII.2.1.3.1.

Comando CREA:

Definición de las propiedades al sistema

Prop	Tipo	Características
CLVE	5	10 dígitos
NMBR	4	30 caracteres
EDAD	1	de 18 a 60 de 1 en 1
SEXO	2	masc o fem
DPTO	3	4 posibles departamentos A, B, C, D.
SLDO	1	5 dígitos (cualquier número comprendido en 5 dígitos)

```
R
#RUNNING 3009
*QUE BASE DE DATOS VAS A UTILIZAR, SI ES UNA NUEVA
DA 0 . EN CASO CONTRARIO DA EL NUMERO DE LA BASE...
#?
0 <---------- SE TRATA DE UNA NUEVA BASE
DAME EL NUMERO DE LA BASE....
20 <---------- SE DA UN NUMERO (CUALQUIERA) COMO IDENTIFICADOR DE LA BASE
DA EL NO DE ARCHIVOS DE LA BASE....
1 <---------- EN ESTA BASE SOLO SE DEFINE UN ARCHIVO

DA EL NOMB. DEL ARCHIVO(4 CAR MAX)
Y EL NUMERO DE IDENTIFICADORES....
AR01 1 <---------- EL ARCHIVO SOLO CONTIENE UN IDENTIFICADOR AL CUAL SE LE CREARA SU TABLA INDICE DE ACCESO DIRECTO
DA EL NOMBRE DEL IDENTIFICADOR LOGICO..
LLA1 <---------- IDENTIFICADOR LOGICO DE LA LLAVE

EL IDENTIFICADOR DEL ARCHIVO ES.... 1 ----> IDENTIFICADOR INTERNO (LO DEFINE EL SISTEMA)

DAME EL NUMERO DE PROPIEDADES QUE TENDRA EL ARCHIVO AR01....
6 <---------- EL ARCHIVO CUENTA CON 6 PROPIEDADES

DA EL NOMBRE DE LA PROPIEDAD(4 CAR MAX) Y TIPO....
LLA1 5 <---------- LA PRIMERA PROPIEDAD ES IDENTIFICADOR (LLA) (TIPO 5)

LA PROPIEDAD ES IDENTIFICADOR DEL ARCHIVO AR01

PROPIEDAD NUEVA, SU IDENTIFICADOR ES 1

DA EL NO. DE DIGITOS MAX DE QUE SE COMPONE LA LLAVE..
6 <---------- Y TIENE 6 DIGITOS COMO MAXIMO

DA EL NOMBRE DE LA PROPIEDAD(4 CAR MAX) Y TIPO....
ARP1 4 <---------- LA SEGUNDA PROPIEDAD ES ALFANUMERICA (AR TIPO=4)

PROPIEDAD NUEVA, SU IDENTIFICADOR ES 2 ----> IDENTIFICADOR INTERNO (DADO POR EL SISTEMA)

DA EL NO. DE CARACTERES MAXIMO QUE PUEDE TENER ESTA PROPIEDAD....
6 <---------- 6 CARACTERES COMO MAXIMO

DA EL NOMBRE DE LA PROPIEDAD(4 CAR MAX) Y TIPO...
ARP2 1 <---------- LA TERCERA PROPIEDAD ES NUMERICA ENTERA (NO TIPO=1)

PROPIEDAD NUEVA, SU IDENTIFICADOR ES 3

DA EL LIMITE INFERIOR, EL SUPERIOR Y
LOS INCREMENTOS DE LOS POSIBLES VALORES DE ESTA PROPIEDAD
ASI COMO LA UNIDAD DE ESTOS VALORES....
1,16,1,DPTOS <---------- HAY 16 DEPARTAMENTOS
```

EJEMPLO COMANDO CREA

```
DA EL NOMBRE DE LA PROPIEDAD(4 CAR MAX) Y TIPO....
A1P3 3 <-------- LA CUARTA PROPIEDAD ES ALFANUMERICA DEFINIDA CAD TIPO = 3)

PROPIEDAD NUEVA, SU IDENTIFICADOR ES  4

DA EL NO. DE VALORES ALFANUMERICOS QUE PUEDE TENERESTA PROP....
2

DA EL VALOR DE LA PRO. SU IDENTIFICADOR DE VALOR ES  1...
MASC <-------- PRIMER VALOR ALFANUMERICO INTERNAMENTE MASC=1

DA EL VALOR DE LA PRO. SU IDENTIFICADOR DE VALOR ES  2...
FEM <-------- SEGUNDO VALOR ALFANUMERICO INTERNAMENTE FEM=2

DA EL NOMBRE DE LA PROPIEDAD(4 CAR MAX) Y TIPO....
A1P4 3 <-------- PROPIEDAD ALFANUMERICA DEFINIDA CAD TIPO =3)

PROPIEDAD NUEVA, SU IDENTIFICADOR ES  5

DA EL NO. DE VALORES ALFANUMERICOS QUE PUEDE TENERESTA PROP....
4

DA EL VALOR DE LA PRO. SU IDENTIFICADOR DE VALOR ES  1...
SOLT

DA EL VALOR DE LA PRO. SU IDENTIFICADOR DE VALOR ES  2...
CASD

DA EL VALOR DE LA PRO. SU IDENTIFICADOR DE VALOR ES  3...
VIUD

DA EL VALOR DE LA PRO. SU IDENTIFICADOR DE VALOR ES  4...
ULIB

DA EL NOMBRE DE LA PROPIEDAD(4 CAR MAX) Y TIPO....
A1P5 2 <-------- PROPIEDAD NUMERICA DEFINIDA

PROPIEDAD NUEVA, SU IDENTIFICADOR ES  6

DA EL NUMERO APROX DE VALORES QUE PUEDE TENER ESTA PRO
ASI COMO LA UNIDAD DE ESTOS VALORES....
10000000000,CTA <-------- HASTA 9 DIGITOS
```

EJEMPLO COMANDO CREA (CONT.)

Ejemplo de PREDI (crea predicado)

SALID (crea salida)

CONSUL (consulta global)

Se desea saber el nombre de los empleados que están en el departamento E Y su número de cuenta es mayor que 400,000

Predicado

P= DPARTAMENTO-2•NO.CTA > 400,000

Salida

S= nombre del empleado

Es importante hacer notar que el predicado inclye a dos propiedades una en R y una en V.

Antes de definir el predicado en términos del sistema si mencionaran las reglas para la formación de un predicado.

Definición de un predicado en el sistema.

En esta primer versión la definición de un predicado es más elemental y laboriosa, pero se puede construir cualquier predicado.

Las reglas son las siguientes:

- se usan identificadores numéricos para asignar variables y operadores (ver tabla siguiente)
- se usa el identificador interno de las propiedades
- el predicado deberá ser expresado en operaciones

elementales (como las mostradas en la tabla)

por ejemplo una expresión lógica y de comparación

$$X\cdot((a>10)+(c\cdot d)) \quad \text{---} \quad (1)$$

a, c, d y X propiedades

primero deberá expresarse en operaciones elementales de la siguiente manera

$A = a>10$ — A, B, C, R, variables

$B = c\cdot d$

$C = A+B$

$R = X\cdot C$ —— (2)

segundo, se deberán usar para las variables identificadores numéricos, así como para los operadores y identificadores internos de propiedades.

	propiedad		identificador
así	a	→	2
	c	→	4
	d	→	8
	x	→	9
	>	→	14
	·	→	5
	+	→	4

y asignado a

A=10

B=11

C=12

R=13

El predicado deberá ser expresado de la forma

var, operador, operando 1, operando 2, operando 3

Var - variable

Operador - ($\cdot$, +, <, >, ... etc)

Operando - propiedad o variable

Así el predicado (2) se tranforma en:

10,14,2, 10,0

11,5,14, 8, 0 —— (3)

12,4,10, 11,

13,5,9, 12

es importante que el último termino de la expresión sea la que opera con todas las demás para obtener el resultado

Como se observa la expresión que requiere el sistema es la expresión placa de (1) manual. En otra versión que incluya un lenguaje de alto nivel deberá transformar una expresión, incluso no como (1) so no de comandos en español a la expresión tipo (3). Para el propósito de la tesis solo se dejo de esa manera.

Indentificador de operador	Operador
1 - $\in$	existencia
2 - $\notin$	no existencia
3 - $\sim$	negación
4 - 0	0 lógico
5 - y	y lógico
6 - OEX	0 exclusivo
7 - P = P	igualdad
8 - P = CTE	"
9 - CTE = P	"
10 - P $\neq$ P	diferencia
11 - P $\neq$ CTE	"
12 - CTE $\neq$ P	"
13 - P > P	mayor que
14 - P > CTE	"
15 - CTE > P	"
16 - P $\geq$ P	mayor-igual
17 - P $\geq$ CTE	"
18 - CTE $\geq$ P	"
19 - P < P	menor que
20 - P < CTE	"
21 - CTE < P	"
22 - P $\leq$ P	menor igual
23 - P $\leq$ CTE	"
24 - CTE $\leq$ P	"

Tabla de Operadores para Consulta
Tabla VII. 2.1.3.1. A

Una vez definido las reglas del predicado transformaremos el predicado de ejemplo

P <= DEPARTAMENTO = 2 . No. CTA. >400,000

identificador de la propiedad departamento - 3 (identificador lógico ARP2).

identificador de la propiedad cuenta = 6 (identificador lógico AOP5).

identificador de operador P = CTE → 8

identificador de operador P > CTE → 14

identificador de operador Y lógico → 5

descomponiendo el predicado

A = DEPARTAMENTO =2

B = No. CTA >400,000

R = A . B

usando identificadores (para A = 11, B=12, C=13) se puede elegir cualquier número

tenemos

11, 8, 3, 2, 0

12, 14, 6, 400,000, 0

13, 5, 11, 12, 0

A continuación se da el listado para el ejemplo de predicado, se harán realmente 3 predicados.

1.- DEPARTAMENTO = 2 para probar la operación solo en V

2.- No. CTA >400,000 para probar operación solo en R

3.- DEPARTAMENTO =2 • No. CTA >400,000 para probar operación en R y V

La salida se genera fundamentalmente con el identificador de propiedad que se desea desplegar (la demás información es de formato de escritura que no fué implementada en esta versión)

Identificador de la propiedad de salida

nombre = 2

Comando SALIDA

Definición de la salida. Se desea imprimir el nombre, de los registros que cumplan. El identificador de dicha propiedad es 2.

El formato de impresión y su ordenamiento, aún no están implementados.

```
*DA COMANDO..PREDIC...(PREDI),SALIDAS(SALID),
             CONSULTA(CONSU),ACTUALIZA(ACTUA),
             DA-CAMPO(DACAPO),RECONFIGURA(RECON),
             FINALIZA(FINAL)....
PREDI
* SI QUIERES GENERAR PR...ADOS, DA IDENTIFICADOR
DE PREDICADO E IDENTIFI...DE ARCHIVO, SI NO DA  -99,-99
 ..............
10,1
*DA VARIABLE,OPERADOR,OPERANDO1,OPERANDO2,OPERANDO3
 (FORMATO LIBRE),SI YA NO HAY MAS DA -99,,,,,,,,
11,8,3,2,0
*DA VARIABLE,OPERADOR,OPERANDO1,OPERANDO2,OPERANDO3
 (FORMATO LIBRE),SI YA NO HAY MAS DA -99,,,,,,,,
-99,,,,,,,,,,,,
* SI QUIERES GENERAR PREDICADOS, DA IDENTIFICADOR
DE PREDICADO E IDENTIFICADOR DE ARCHIVO, SI NO DA  -99,-99
 ..............
20,1
*DA VARIABLE,OPERADOR,OPERANDO1,OPERANDO2,OPERANDO3
 (FORMATO LIBRE),SI YA NO HAY MAS DA -99,,,,,,,,
12,14,6,400000,0
*DA VARIABLE,OPERADOR,OPERANDO1,OPERANDO2,OPERANDO3
 (FORMATO LIBRE),SI YA NO HAY MAS DA -99,,,,,,,,
-99,,,,,,,,,,,,,
* SI QUIERES GENERAR PREDICADOS, DA IDENTIFICADOR
DE PREDICADO E IDENTIFICADOR DE ARCHIVO, SI NO DA  -99,-99
 ..............
30,1
*DA VARIABLE,OPERADOR,OPERANDO1,OPERANDO2,OPERANDO3
 (FORMATO LIBRE),SI YA NO HAY MAS DA -99,,,,,,,,
11,8,3,2,0
*DA VARIABLE,OPERADOR,OPERANDO1,OPERANDO2,OPERANDO3
 (FORMATO LIBRE),SI YA NO HAY MAS DA -99,,,,,,,,
12,14,6,400000,0
*DA VARIABLE,OPERADOR,OPERANDO1,OPERANDO2,OPERANDO3
 (FORMATO LIBRE),SI YA NO HAY MAS DA -99,,,,,,,,
13,5,11,12,0
*DA VARIABLE,OPERADOR,OPERANDO1,OPERANDO2,OPERANDO3
 (FORMATO LIBRE),SI YA NO HAY MAS DA -99,,,,,,,,
-99,,,,,,,,,,,,
* SI QUIERES GENERAR PREDICADOS, DA IDENTIFICADOR
DE PREDICADO E IDENTIFICADOR DE ARCHIVO, SI NO DA  -99,-99
 ..............
-99,,,,,
```

```
PREDICADO       10        1        1        1
     8      3       2        0        1

PREDICADO       20        1        2        2
    14      6  400000        0        1

PREDICADO       30        1        3        5
     8      3       2        0        1
    14      6  400000        0        1
     5     -3      -4        0        3
```

← ESTRUCTURA INTERNA DE LOS PREDICADOS

COMANDO PREDICADO

```
*DA COMANDO..PREDICADOS(PREDI),SALIDAS(SALID),
             CONSULTA(CONSU),ACTUALIZA(ACTUA),
             DA-CAMPO(DACAPO),RECONFIGURA(RECON),
             FINALIZA(FINAL)....
SALID
*DAME IDENTIFICADOR DE LAFUNCION DE SALIDA
 IDENTIFICADOR DEL ARCHIVO A OPERAR
 SI YA NO HAY MAS DA -99,,,,,,,,,,,,,,,
11,1
*DA ID. DE PROPD.,NO. DE CAMPOS,FORMSTO,ESPACIOS
 SI YA NO HAY MAS DA -99,,,,,,,,,,
2,6,1,1
*DA ID. DE PROPD.,NO. DE CAMPOS,FORMSTO,ESPACIOS
 SI YA NO HAY MAS DA -99,,,,,,,,,
-99,,,,,,,
*DA EL ARCHIVO DE IMPRESION..
6
*QUIERES LA INFORMACION ORDENADA..
NO
*DAME IDENTIFICADOR DE LAFUNCION DE SALIDA
 IDENTIFICADOR DEL ARCHIVO A OPERAR
 SI YA NO HAY MAS DA -99,,,,,,,,,,,,,,,
-99,,,,,,,
        1       1       1       6
    2       6       1       1       0      21      48
```

COMANDO SALIDA

Comando DACPO (acceso directo)

El uso de este comando es muy fácil, se da el identificador del archivo, el identificador lógico del registro a accesar y finalmente la propiedad que se desea consultar.

A continuación se ve un ejemplo de consulta en R y uno de consulta en V con acceso directo.

a) Se desea conocer el número de aumento del trabajador 1515 (su identificador interno es 37). En este caso se accesa directamente una propiedad en R.

b) Se desea saber el departamento del mismo empleado. acceso directo en V.

```
*DA COMANDO..PREDICADOS(PREDI),SALIDAS(SALID),
             CONSULTA(CONSU),ACTUALIZA(ACTUA),
             DA-CAMPO(DACAPO),RECONFIGURA(RECON),
             FINALIZA(FINAL)....
DACPO
DA ID. DE ARCHIVO,
SI YA ES TODO DA -99.........
1
DA ID. DE IDENTIFICADOR,REG A BUSCAR Y
EL IDENTIFICADOR DE PROP. A CONSULTAR.....
1,1515,6

LA PROPIEDAD     6 DEL REGISTRO NO.      37 ESTA EN R


EL REG    1515    TIENE EN LA PROP      6
EL VALOR 299937   1


DA ID. DE ARCHIVO,
SI YA ES TODO DA -99.........
1
DA ID. DE IDENTIFICADOR,REG A BUSCAR Y
EL IDENTIFICADOR DE PROP. A CONSULTAR.....
1,1515,3
LA PROPIEDAD     3 DEL REG. NO      37ESTA EN V


EL REG    1515    TIENE EN LA PROP      3
EL VALOR       4
DA ID. DE ARCHIVO,
SI YA ES TODO DA -99.........
-99,,,
```

COMANDO DACPO

Comando Consulta:

Una vez definidos predicados y salidas el comando consulta puede usar cualquier combinación entre ellos.

La información que requiere este comando es la siguiente:

1.- Identificador de archivo a operar

2.- Identificador de predicado

3.- Identificador de salida

4.- Identificador de vector de entrada (subconjunto del archivo sobre cual se desea operar)

5.- Identificador de vector de salida (subconjunto que se creará al aplicar el predicado al subconjunto)

6.- Archivo donde se desea que el vector de salida apunte (puede ser a otro archivo, si el primero contiene como propiedad el identificador del segundo).

7.- Identificador de propiedad llave que se usará para la proyección (en principio un archivo puede tener varias llaves de acceso).

Notas sobre algunos valores de los argumentos de consulta.

- el usuario puede usar hata 6 identificadores numéricos (del 1 al 6) para vectores.

- la salida puede tener vabres

  1.- Se produce vector de salida pero no impresión (para operaciones entre archivo).

  0.- Se produce vector de salida y solo se imprime los identificadores internos de los registros que cumplieron.

  >0. Se produce vector de salida y se genera una salida.

- si el predicado no se aplica a un subconjunto si no a todo el archivo el identificador de vector de entrada debe ser = 0

- si la proyección es sobre el mismo archivo, el archivo de proyección e identificador de proyección deberá ser 0.

```
#DA COMANDO .PREDICADOS(PREDI),SALIDAS(SALID),
             CONSULTA(CONSU),ACTUALIZA(ACTUA),
             DA-CAMPO(DACAPO),RECONFIGURA(RECON),
             FINALIZA(FINAL)....
CONSUL
#DA IDENTIFICADOR DE ARCHIVO.DE PREDICADO.DE SALIDA
 IDENTIFICADOR DE VECTOR DE ENTRADA. DE VECTOR DE SALIDA. SOBRE
 QUE ARCHIVO SE PROYECTARA LA SALIDA
Y EL IDENTIFICADOR DE LA PROP. DE PROYECCION
(SI QUIERES FIN DA '-99'.....................
1.1.1.0.1.5.0
  TIPO DE PREDICADO Y DE SALIDA      2      3
PROPIEDADES QUE CUMPLIERON CON EL PREDICADO..

  15 ******
             NFU414
  25 ******
             NU0INM
  30 ******
             NM5NP0
  31 ******
             N2007E
  32 ******
             NTHAL0
  34 ******
             N27211
  39 ******
             NK36KR
  59 ******
             NREFLP
  53 ******
             N0NT66
  80 ******
             N6HY9N
```

COMANDO CONSUL

```
*DA IDENTIFICADOR DE ARCHIVO,DE PREDICADO,DE SALIDA
 IDENTIFICADOR DE VECTOR DE ENTRADA, DE VECTOR DE SALIDA, SOBRE
 QUE ARCHIVO SE PROYECTARA LA SALIDA
Y EL IDENTIFICADOR DE LA PROP. DE PROYECCION
(SI QUIERES FIN DA  -99),,,,,,,,,,,,,,,,,,,,,
1,2,0,0,3,0,0
  TIPO DE PREDICAD Y DE SALIDA      1      0
  PROPIEDADES QUE CUMPLIERON CON EL PREDICADO..


********************************** ----     1
********************************** ----     3
********************************** ----     4
********************************** ----    10
********************************** ----    13
********************************** ----    15
********************************** ----    16
********************************** ----    17
********************************** ----    18
********************************** ----    28
********************************** ----    29
********************************** ----    56   <-- EN ESTE CASO SOLO SE IMPRIMIERON LOS
********************************** ----    60       IDENTIFICADORES (SALIDA = 0)
********************************** ----    65
********************************** ----    66
********************************** ----    67
********************************** ----    70
********************************** ----    73
********************************** ----    75
********************************** ----    82
********************************** ----    89
********************************** ----    90
********************************** ----    99
*DA IDENTIFICADOR DE ARCHIVO,DE PREDICADO,DE SALIDA
 IDENTIFICADOR DE VECTOR DE ENTRADA, DE VECTOR DE SALIDA, SOBRE
 QUE ARCHIVO SE PROYECTARA LA SALIDA
Y EL IDENTIFICADOR DE LA PROP. DE PROYECCION
(SI QUIERES FIN DA  -99),,,,,,,,,,,,,,,,,,,,,
1,3,1,0,3,0,0
  TIPO DE PREDICAD Y DE SALIDA      3      3
PROPIEDADES QUE CUMPLIERON CON EL PREDICADO..


 15 ******
          NPV414
*DA IDENTIFICADOR DE ARCHIVO,DE PREDICADO,DE SALIDA
 IDENTIFICADOR DE VECTOR DE ENTRADA, DE VECTOR DE SALIDA, SOBRE
 QUE ARCHIVO SE PROYECTARA LA SALIDA
Y EL IDENTIFICADOR DE LA PROP. DE PROYECCION
(SI QUIERES FIN DA  -99),,,,,,,,,,,,,,,,,,,,,
-99,,,,,,,,,,,,,
```

COMANDO CONSUL (CONT)

Comando ACTUA (cambios de valores en las propiedades).

También es una operación sencilla como la de DALPO, se requiere del identificador del archivo a operar el identificador de propiedad a modificar y el nuevo valor actualizar.

Se darán dos ejemplos de ACTUA en R y en V y finalmente para comprobar si el cambio de valores en las propiedades se realizó bien, se aplicará el comando DACPO.

Se desea cambiar el departamento del empleado 1515 a 7 (actualización en V) y su número de cuenta a 99 (actualización en R).

```
*DA COMANDO..PREDICADOS(PREDI),SALIDAS(SALID),
             CONSULTA(CONSU),ACTUALIZA(ACTUA),
             DA-CAMPO(DACAPO),RECONFIGURA(RECON),
             FINALIZA(FINAL)....
ACTUA
*SOBRE QUE ARCHIVO QUIERES ACTUALIZAR...
 SI ES TODO DA -99.........
1
*DA EL IDTF INTERNO DEL IDTF  QUE VAS A USAR,
SU VALOR (= AL REG QUE QUIERES MODIFICAR
 Y EL NO. DE PROPIEDADES QUE VAS A MODIFICAR.....
SI ES TODO DA -99,.........
1,1515,2
*DA IDENTIFICADOR DE PROP. Y SU NUEVO VALOR..
3,7
*DA IDENTIFICADOR DE PROP. Y SU NUEVO VALOR..
6,99
     99       69       30
SE MODF EL REG..     37  EN R
SE MODF EL REG..     37 EN V
*SOBRE QUE ARCHIVO QUIERES ACTUALIZAR...
 SI ES TODO DA -99.........
-99,,,
```

COMANDO ACTUA

```
*DA COMANDO..PREDICADOS(PREDI),SALIDAS(SALID),
             CONSULTA(CONSU),ACTUALIZA(ACTUA),
             DA-CAMPO(DACAPO),RECONFIGURA(RECON),
             FINALIZA(FINAL)....
DACPO
DA ID. DE ARCHIVO,
SI YA ES TODO DA -99.........
1
DA ID. DE IDENTIFICADOR,REG A BUSCAR Y
EL IDENTIFICADOR DE PROP. A CONSULTAR.....
1,1515,3
LA PROPIEDAD     3 DEL REG. NO      37ESTA EN V


EL REG   1515   TIENE EN LA PROP      3
EL VALOR      7
DA ID. DE ARCHIVO,
SI YA ES TODO DA -99.........
1
DA ID. DE IDENTIFICADOR,REG A BUSCAR Y
EL IDENTIFICADOR DE PROP. A CONSULTAR.....
1,1515,6

LA PROPIEDAD     6 DEL REGISTRO NO.      37 ESTA EN R


EL REG   1515   TIENE EN LA PROP      6
EL VALOR      99
DA ID. DE ARCHIVO,
SI YA ES TODO DA -99.........
-99,,,
```

COMANDO DACPO PARA VERIFICAR COMANDO ACTUA

VII.2.1.3.2 Funciones especiales con comandos básicos

Este aspecto del lenguaje a primer nivel es muy interesante y está orientado a gente con experiencia en programación y con conocimiento de la base implementada al sistema (la utilización de los comandos básicos simplemente no requiere del usuario, conocimientos de programación).

Estas funciones no son específicas y dependerán del tipo de aplicación especial de la base de datos en particular implementado al sistema.

De esta manera se pueden pensar en funciones que consulten periódicamente archivos y por ciertas condiciones modifiquen los archivos (actualizaciones) o que informen al usuario de cambios importantes en los datos de la base. También puede pensarse en desarrollar un subsistema para llevar estadísticas automáticas del comportamiento de los datos en cuanto al sistema que representan con el objeto de hacer proyecciones simulaciónes, etc.

Por ejemplo, supóngase que se tiene un archivo de inventarios y que cada cierto tiempo se desea rebajar en un 10% el valor de la mercancía que tenga un cierto tiempo de estancia en la bodega y si su cantidad es menor de cierto volúmen. Esta función se puede hacer automática. Se puede hacer que corra cada vez que se tenta que hacer la revisión del estado

de la mercancía en la bodega. Con los siguientes comandos básicos y funciones elementales y un poco de programación puede realizarse tal función especial como es mostrado como sigue:

```
     PREDI=FCHI < FCH1   y CTD < VOL1
     SALD1= ARTC, CTD
     CONSUL (ARTICULOS, PREDI, SALD1, 0, VS, ATCLS)
10   IF(DIR=DADIR(VS)) RETURN
     VAL=DAPROP (ARTICULOS, ATCLS=DIR, COSTO,E) *.90
     MTPROP (ARTICULOS, ATCLS=DIR, CSR, VAL, Z)
          GO TO 10
```

En la primera instrucción se especifica la condición de acceso al archivo de artículos (predicado), La propiedad fecha de ingreso de la última remesa (FCHI) tiene que ser menor de una fecha especificada, y también se debe cumplir la propiedad cantidad en existencia (CTD) debe ser menor que cierto volúmen especificado (VOLI).

En la segunda instrucción se define la información que se requiere de los artículos que hayan cumplido con la condición, en este caso se pide el nombre del artículo y su cantidad en existencia.

En la tercera instrucción se hace el acceso global al archivo de artículos con el predicado y salida antes especificado.

No se dá vector de entrada, lo que en esta función significa considerar a todo el archivo.

Como resultado se obtendrá el vector de acceso que apunta a los identificadores ATCLS que son en este caso, del archivo ARTICULOS.

En la cuarta instrucción, se obtiene del vector de acceso la próxima dirección del registro que cumple con la condición impuesta por el predicado.

En la quinta instrucción se obtiene y opera el valor de la propiedad accesada a través de la dirección dada por el vector de acceso.

En la sexta instrucción se actualiza el acceso valor del costo.

Finalmente se continúa con todos los artículos de la misma manera antes señalada.

Esto es solo un ejemplo de las posibilidades que permite el uso de los comandos básicos, funciones elementales y un poco de lenguaje de programación.

Esta parte del lenguaje de primer nivel es muy importante, puesto que permitirá una explotación más amplia de una base de datos.

En la tercera instrucción se hace el acceso global al archivo de artículos con el predicado y salida antes especificado.

No se dá vector de entrada, lo que en esta función significa considerar a todo el archivo.

Como resultado se obtendrá el vector de acceso que apunta a los identificadores ATCLS que son en este caso, del archivo ARTICULOS.

En la cuarta instrucción, se obtiene del vector de acceso la próxima dirección del registro que cumple con la condición impuesta por el predicado.

En la quinta instrucción se obtiene y opera el valor de la propiedad accesada a través de la dirección dada por el vector de acceso.

En la sexta instrucción se actualiza el acceso valor del costo.

Finalmente se continúa con todos los artículos de la misma manera antes señalada.

Esto es solo un ejemplo de las posibilidades que permite el uso de los comandos básicos, funciones elementales y un poco de lenguaje de programación.

Esta parte del lenguaje de primer nivel es muy importante, puesto que permitirá una explotación más amplia de una base de datos.

VIII OBSERVACIONES

En general el sistema en la versión actual, no permite una gran facilidad para la implementación de sistemas reales, sus principales carencias, que se podrían resumir en la falta de lenguaje de alto nivel y orientado a la base de datos son:

- Manejo de identificadores internos para las propiedades, en lugar de identificadores lógicos.
- la formación de predicados muy elemental y complicados para uso corriente
- formateador para las salidas

No obstante están desarrolladas las funciones básicas, que en una versión posterior podrían desarrollarse los subsistemas que podrían permitir un uso más fácil del sistema.

Por otro lado, hay que señalar con las restricciones que cuenta la primera versión y que puede resumirse como sigue:

- la historia del comportamiento usa un esquema sencillo (frecuencias de operación acumuladas)
- el método de evaluación para la reconfiguración, fundamentalmente es sobre propiedad, con un factor correctivo.

Estas restricciones en esta versión, pueden dar no la

mejor solución al problema de la reconfiguración, no obstante, en versiones posteriores podrán emplearse otros métodos de llevar al comportamiento y de su evaluación sin afectar básicamente la estructura actual.

IX CONCLUSIONES

En general, el problema que en un momento dado puede plantear un sistema con reconfigurador, es el costo que este subsistema en sí implica (costo en llevar contínuamente el comportamiento del sistema y el costo en realizar, en ciertos períodos de tiempo, la evaluación para una reconfiguración). De tal suerte que su operación tendrá que ser lo más simple posible para que la reconfiguración realmente cumpla con su objetivo, optimización en el tiempo de respuesta y recursos de cómputo.

El tiempo y esfuerzo que tomó desarrollar toda la estructura de un sistema general, no permitió, por falta de tiempo, la evaluación entre varios métodos de reconfiguración. Por otro lado (por razones también de tiempo y por no contar con una versión del sistema con uso más sencillo) no fué posible poder usar base de datos reales y de grandes volúmenes que permitieran a través de su uso, tener más elementos para la evaluación del reconfigurador.

De esta manera no se puede decir que el método de reconfiguración aquí usado es uno de los mejores. Pero si se puede decir que la ventaja de un reconfigurador en general es la optimización, y su desventaja, el precio que significa tener el reconfigurador, el cual sólo tendrá sentido cuando el costo ganado en la optimización sea mayor que el costo

pagado por tener el reconfigurador.

Por otro lado, hay que señalar que la reconfiguración apunta sólo sobre una parte del problema de grandes almacenamientos, diferentes tipos de estructuras y altos tiempos de acceso. En este caso fué una solución por 'software' al acceso de datos en memoria secundaria de acceso directo. Existen otras alternativas como son el empleo de memoria principal y no de secundaria (el costo de memoria principal tiende a ser muy reducido, como el caso de memorias de burbuja) o una solución por 'hardware' con implementaciones que permitan accesos rápidos.

En términos generales este trabajo ha servido para desarrollar e implementar toda una estructura básica para elaborar una base de datos general y reconfigurable, dado el trabajo que sólo esto representa; y no ha sido un objetivo presentar las técnicas más óptimas de reconfiguración.

Sin embargo, la función de reconfiguración puede ser cambiada por otras mejores sin que esto altere la estructura básica del sistema.

De igual manera se puede decir en cuanto al acabado 'fino' del sistema; programas adicionales que permitan una facilidad en el diálogo sistema usuario, por ejemplo el lenguaje orientado al manejo de datos y en español.

No obstante, todos estos "detalles" pueden realizarse en un futuro sin que por esto se altere la estructura básica.

Finalmente solo resta decir que el sistema fué programado en FORTRAN y en Sistema Burroughs B6700 (del CSC-UNAM).

Básicamente el sistema fué programado en FORTRAN-IV excepto algunas partes, bien específicas (algunas operaciones y manejos de archivos) en FORTRAN de Burroughs. De esta manera, incluso este sistema puede ser transportado a minicomputadoras, teniendo solo que adaptar la parte de operaciones y manejo de archivos (que representa un porcentaje pequeño comparado con la totalidad del sistema) y la definición del tamaño de memoria disponible para operación y tamaño de palabra del sistema.

X BIBLIOGRAFIA

1. An introduction to data-base system
   C.J. Date, Addison Wisley
2. Principles of data base management
   J. Martin. Prentice Hall
3. Information system: Theory and Practice
   Burch and Strater. Hamilton
4. Design of on-line computer system
   E. Gourdon. Prentice Hall
5. Data-Base
   James Martin. Prentice Hall
6. Management Data-Base
   R. Clay Sprowls. Wiley/Hamilton
7. Data Base management systems
   Donald A. Jardine. North -Holland American Elsevier
8. Sistemas de base de datos
   EXIR, CICS, Banco Geográfico, Aladin.