

DISEÑO DEL SISTEMA OBJETO MULTINODAL ORIENTADO A DISCO PARA BASES DE DATOS

EDSCOTT WILSON GARCIA

Subdirección de Tecnología de
Refinación y Petroquímica del
IMP

En este artículo se describe la metodología empleada para el diseño de una base de datos con una estructura ramificada y de número de ramas por nodo variable. La finalidad de dicho diseño es el de proveer una base de datos que pueda manejar un volumen excesivo de información con rapidez y confianza. Se describen los lineamientos generales del diseño, las fórmulas matemáticas de su desempeño y las conclusiones obtenidas en su aplicación.

INTRODUCCION

En los últimos años, el avance que ha tenido la computación la ha situado en uno de los puntos más dinámicos de la investigación. Uno de los desafíos más interesantes que presenta, por los problemas inherentes, son las estrategias en el diseño de bases de datos.

Se debe recordar que la capacidad que tiene la computadora para trabajar sobre ecuaciones difíciles una y otra vez sin equivocarse, la han convertido en el principal aliado de los físicos, químicos teóricos e ingenieros de distintas ramas.

La computación ha abierto alternativas nunca antes sospechadas en las áreas de investigación. Antes del advenimiento de los microcircuitos y de los medios magnéticos de almacenaje, no se podía trabajar con grandes volúmenes de datos sin perder la perspectiva de lo que se estaba estudiando. Era necesario reducir aquella información mediante fórmulas generales, ya fueran rectas o curvas, o bien hacer un análisis estrictamente estadístico.

Por otro lado, el control de procesos químicos que requiere la industria o el monitoreo de variables meteorológicas en una área urbana que sufra los problemas de la contaminación atmosférica, son situaciones donde la información generada es tan voluminosa que son capaces, no tan sólo de abrumar al investigador

más tenaz, sino que fácilmente pueden saturar cualquier sistema de cómputo.

¿Pero de qué sirve tener tanta información?, se preguntará el lector. Quizás el obtener esta cascada de números parezca, a primera vista, pecar de puntillosos, pero no es así. Para la toma de decisiones urgentes es extremadamente importante la rapidez con que se actúe. Es más razonable ver en el monitor de una estación de trabajo una representación gráfica de los patrones meteorológicos y la subsecuente dispersión de contaminantes, y así identificar las áreas críticas, que recibir una avalancha de papel con números y números y más números, que después de cierto tiempo pierden todo sentido.

Para manejar esta información, ya sean los datos de operación de veinte distintas variables que se registran electrónicamente en la operación de una planta piloto, o las características de miles de gasóleos analizados por los laboratorios petroleros o los registros de un monitoreo multivariable de los contaminantes atmosféricos, es necesario una organización estructurada que minimice el tiempo de búsqueda y acceso.

Uno de los problemas con los cuales se enfrenta el programador, y quizás el más importante desde el punto de vista de la informática, es éste. Si bien hay muchas maneras de guardar la información, no todas son óptimas. Como punto de partida se deben

tomar en cuenta dos consideraciones importantes; la primera es que la información deberá ocupar el menor espacio posible y, segundo, debe tener un tiempo de acceso mínimo.

El monitoreo de una planta de proceso es un ejemplo donde el desarrollo de nuevas tecnologías se ve limitado por la capacidad de observación. Para resolver esto es necesario contar con un sistema que reciba los datos, los guarde, los organice y sea capaz de generar imágenes que representen la situación en cuestión de instantes. Además, se debe poder volver atrás en el tiempo y estudiar con mayor detalle cualquier momento de interés.

La cuestión de generación de imágenes es difícil, así como la comunicación entre aparatos y la captura de información. Pero el meollo del asunto está en la organización: dicho de otra forma, en la estructura y el diseño de la base de datos.

Para clarificar lo expuesto, se tiene la siguiente situación:

En el estudio de una planta piloto se hace un arranque y se monitorea el comportamiento durante un intervalo, variando las condiciones de operación. Se toman lecturas y se comparan.

Este método de estudio tiene la desventaja de estudiar un proceso dinámico con variación respecto al tiempo (dt), en forma estática. Para hacer un estudio más real y ver cómo las variaciones en los parámetros de operación afectan la evolución del proceso, se tiene que recopilar información sobre los parámetros representativos y analizar su variación con respecto al

tiempo (d/dt), aproximando con una Δt suficientemente pequeña. Esta Δt representa, en términos reales, el monitoreo de forma automática de las variables medidas de la operación de la planta.

Esta información tendría que ser alimentada a un sistema de cómputo y dirigida hacia un fichero de disco. Para monitorear la operación, la información debe ser procesada mediante una interfase gráfica.

ANTECEDENTES PARA EL DISEÑO

Existen a la fecha dos métodos con los cuales se maneja información en memoria dinámica². En primer lugar se tienen las listas ligadas y, en segundo, las listas con estructura de árbol (véase la figura 1).

Para acceder un elemento en particular, se debe comenzar por el principio y recorrer la lista hasta encontrar el elemento requerido.

Los archivos secuenciales de disco funcionan de esta forma y se debe notar que es necesario leer $n-1$ elementos antes de llegar al elemento n .

Existe, no obstante, otra forma de archivos, denominados de acceso aleatorio. En este tipo de archivo se puede acceder directamente al registro n sin tener que leer los anteriores. En cierta forma, el utilizar este tipo de archivos es como si se tuviera un arreglo definido en memoria dinámica.

El problema con esto es que no hay forma de saber a qué n corresponde el elemento que se desea encontrar.

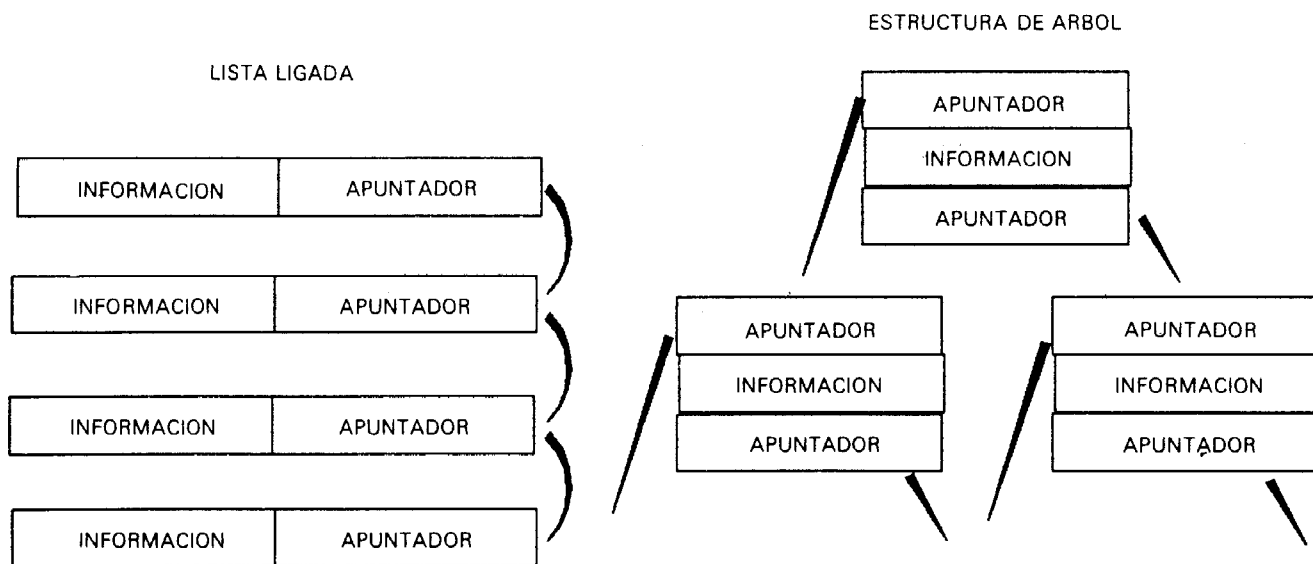


Fig. 1.— Tipos de estructura de memoria.

Para coadyuvar hacia la solución de este problema, algunos sistemas han implantado los archivos indexados, donde se genera un índice de llaves que permite hacer una lectura en una posición más cercana al dato requerido.

El problema que surge con los procesos dinámicos, es que los archivos deben estar creciendo constantemente y, por lo tanto, esta solución no es factible ya que mientras más crece el archivo, más se tarda en añadir nuevos elementos, llegando a un punto en que el tiempo es tanto que se hace impracticable en su aplicación¹.

El segundo tipo de organización de memoria es el de árbol. En este, cada registro, o nodo, apunta a dos o más nodos.

Para diseñar una base de datos con esta mentalidad, pero orientándolo a disco, se deben tomar en cuenta las siguientes consideraciones.

1. Para que sea general y pueda ser utilizado en programación orientada a objetivos, debe ser variable el número de ramas por nodo.
2. Al definir el nodo más alto, deben quedar definidos los nodos inferiores.
3. Los nodos deben ir reduciendo el número de ramas al ir disminuyendo su nivel para lograr un aprovechamiento al máximo del espacio de disco.
4. El tamaño de cada registro deberá ser variable para su aplicación en problemas de diversa índole y, otra vez, para aprovechar al máximo la capacidad de almacenaje de disco.
5. Se deberán añadir, modificar, o borrar registros de la base de datos.
6. Se deberá hacer un recorrido de los elementos de una forma rápida y sencilla³.

SISTEMA OBJETO MULTINODAL ORIENTADO A DISCO

La estructura SOMOD parte de un principio de nodos multirama donde el número de éstos varía según el nivel de profundidad. El nodo de tipo superior tiene n ramas, mientras que todos los nodos localizados sobre la rama 1 de este nodo superior tienen a su vez n ramas (véase la figura 2)

Los nodos que se encuentran en la rama 2 tendrán $n-1$ ramas y ninguna de ellas será del tipo 1, mientras que los nodos que se encuentran en la rama 3 tendrá $n-2$ ramas, prescindiendo de las ramas de tipo 1 y de tipo 2 (véase la figura 3).

De esta forma se va reduciendo consecutivamente el número de ramas por nodo hasta llegar a los que se

encuentran sobre el tipo $n-1$. Estas tendrán sólo dos ramas que serán de tipo $n-1$ y n (véase la figura 4).

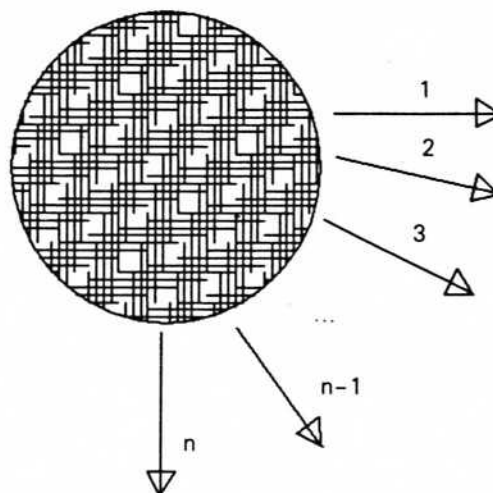


Fig. 2. — Nodo de tipo 1.

Por último, los nodos sobre la rama n sólo tendrán una rama, que será de tipo n .

¿Pero por qué disminuye el número de ramas? Un análisis detallado indica que mientras más se profundice en la estructura, se requiere reducir la complejidad del nodo para así mantener el tamaño de cada registro al menor posible.

En cuanto a la organización, ésta se hace mediante la codificación de llaves de acceso. Si en el nodo superior se tienen n ramas, se tendrá para cada uno de los nodos de la base de datos, una llave compuesta de n partes.

Ningún nodo puede tener una llave idéntica a la de otro nodo. Para formar una llave de nueve caracteres, a partir de datos que representen —por ejemplo— la hora, el día, el mes y el año, lo primero que se debe hacer es expresar el tiempo relativo a una fecha de inicio. A partir de entonces, a cada hora, día, mes y año corresponderá un número único de horas transcurridas. En un periodo de diez años, el número máximo será $24 \times 365.25 \times 10 = 876609$. Para elaborar una llave de nueve caracteres, basta expresar este número en base tetradecimal: 111121230.

Ahora bien, cada uno de estos dígitos representará la dirección sobre cada una de las ramas del nodo superior. Dicho de otra forma, a cada rama corresponde un dígito, tal que todos los nodos que se encuentran al ir por la rama 1 tienen el dígito 1 distinto. En un sistema tetradecimal, esto significa una longitud máxima de cuatro nodos por rama.

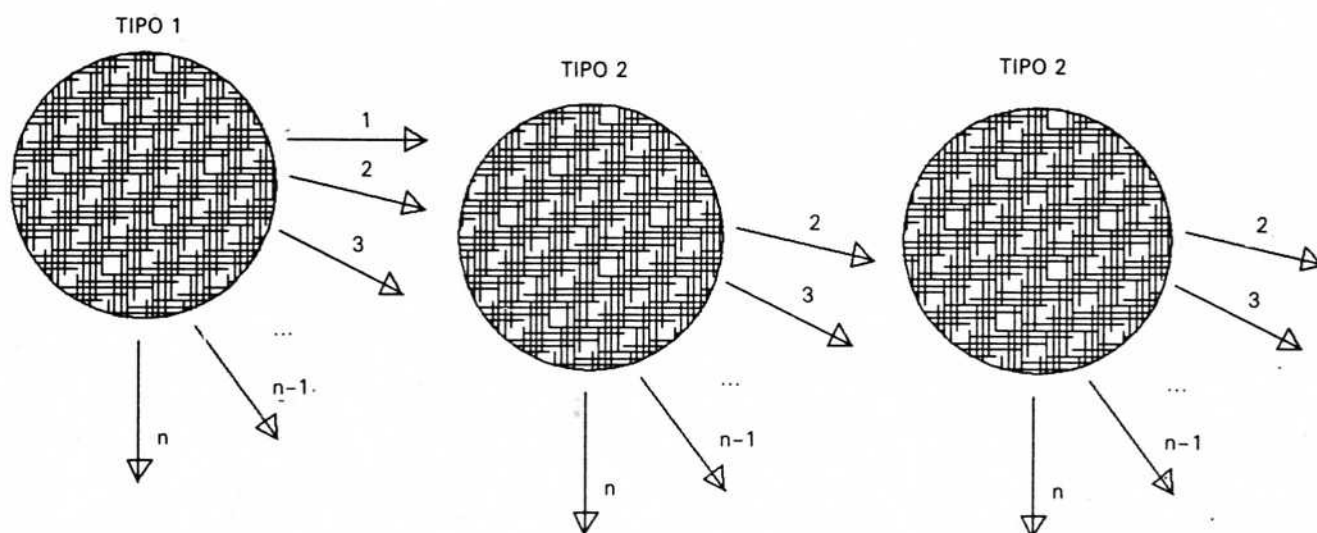


Fig. 3.— Nodos de tipo 1 y 2.

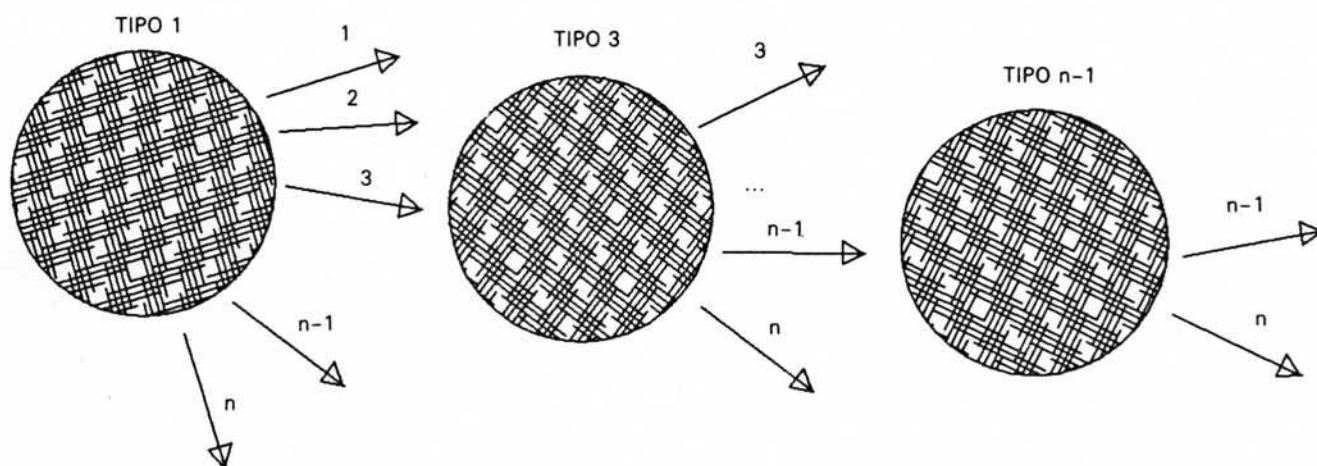


Fig. 4.— Nodos de tipo 1, 3 y (n-1).

Al bajar por la rama 2, todos los elementos tendrán el dígito 1 igual y variará el segundo dígito. Siguiendo este patrón se deduce que al recorrer la rama n , todos los dígitos desde el primero hasta el $n-1$ de la llave serán iguales. Se sigue el procedimiento hasta encontrar el nodo en que todos los dígitos de la llave sean iguales a aquellos del elemento que se desea encontrar.

Un sencillo razonamiento indica que, en el peor de los casos, para encontrar un elemento de entre los 87660, se tendrá que recorrer la longitud de cada una de las nueve ramas. Es decir, 36 interrogantes antes de encontrar lo que se busca.

Un análisis matemático de la estructura SOMOD da como resultado que el número máximo de lecturas que tiene que realizar, siguiendo el procedimiento de formación de llaves anteriormente descrito, está dado por la función ($F(J)$):

$$F(J) = J \times m^{1/J} \quad (1)$$

que tiene un valor mínimo (J) en

$$J = \ln(m) \quad (2)$$

Donde J es el número de llaves que minimizan las lecturas en el peor de los casos y m es el número de datos. Para dos millones de datos, $J \approx 14$.

Pero la variación cerca del mínimo es muy poca y, desde un punto de vista práctico, no conviene llegar al mínimo sino sólo acercarse. Para esto se recurrió a un criterio ingenieril. Se aumentó el número de llaves hasta que la diferencia de uno reporta una variación menor al diez por ciento. En el caso antes mencionado, diez llaves serían suficientes para un buen desempeño del sistema.

En el caso de un archivo con dos millones de registros, se tendría un número de 43 lecturas como máximo.

Otro factor importante es calcular el tamaño que representa dicha estructura para la base de datos.

Reduciéndolo a términos generales, la expresión queda:

$$\text{bytes} = 4 \sum_{n=1}^J (J-n+1)(V^n - V^{n-1}) \quad (3)$$

donde:

V = Valor máximo por llave

J = Número de ramas.

El sistema SOMOD se ha aplicado en el Instituto Mexicano del Petróleo para el desarrollo de un sistema de visualización de factores meteorológicos y de distribución de contaminantes sobre el área metropolitana del Valle de México. También, se ha utilizado para el cálculo estadístico de esta misma información, produciéndose resultados notables en cuanto a su eficiencia.

Debido a estos resultados y a los desarrollos que se continúan realizando alrededor del sistema SOMOD se puede concluir que es un notable avance en cuanto al diseño de bases de datos para el manejo eficiente de la información. Esto repercutirá en un mejor entendimiento de los procesos y fenómenos que sean susceptibles a un examen riguroso de extracción de información.

REFERENCIAS

1. Abelson, Sussman: "Structure and Interpretation of Computer Programs", Cambridge, Mass., MIT Press (1985).
2. Berry: "C + + Programming"; Carmel, IN. SAMS (1992).
3. Wasserman, P.: "Object-oriented structured design and C + +", Computer Language, 8(1):41-52 (1991).