(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 326 657**

(51) Int. Cl.:
G06F 17/30 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Número de solicitud europea: **04787404 .5**
(96) Fecha de presentación : **21.09.2004**
(97) Número de publicación de la solicitud: **1700233**
(97) Fecha de publicación de la solicitud: **13.09.2006**

(54) Título: **Procedimiento de organización de una base de datos.**

(30) Prioridad: **22.09.2003 FR 03 10474**

(45) Fecha de publicación de la mención BOPI:
16.10.2009

(45) Fecha de la publicación del folleto de la patente:
16.10.2009

(73) Titular/es: **KoDe**
63, rue de la Colonie
75013 Paris, FR

(72) Inventor/es: **Koskas, Joseph Michel**

(74) Agente: **Arias Sanz, Juan**

DESCRIPCIÓN

Procedimiento de organización de una base de datos.

La presente invención se refiere al campo de las bases de datos. La presente invención se refiere, más en particular, a un procedimiento técnico de organización y de tratamiento de consultas de una base de datos.

En la técnica anterior se conoce, a partir de la solicitud de patente estadounidense US 2004/0098363 (IBM), un sistema de almacenamiento jerárquico de datos. Se almacenan objetos de datos con una jerarquía de almacenamiento y se generan tablas de contenido que contienen entradas. La ubicación de las tablas de contenido se gestiona de manera dinámica.

En la técnica anterior se conocen también, a partir de la solicitud de patente europea EP 1 423 799 (Lafayette Software) procedimientos para organizar datos y realizar consultas en un sistema de bases de datos. La información está organizada en un sistema de bases de datos con grupos de atributos definidos y palabras de recopilación de datos asignadas a los atributos asociando una lista de identificadores de gráficos de datos con una entrada de tesauro.

En la técnica anterior se conoce, a partir de la solicitud US 2003/130981 A1 (Nehru Archana [US] *et al*) un procedimiento de construcción de árboles de radicales (*radix trees*) que ahorran espacio de almacenamiento. En la técnica anterior se conoce también, a partir de la solicitud US-A-5 826 262 (Bui Thuang Quang [US] *et al*) un procedimiento de construcción de árboles de radicales paralelizado.

En la técnica anterior se conoce también, a partir de la solicitud de patente PCT WO 04/25507 (Karmic Software Research), que corresponde a la solicitud de patente francesa FR 2 844 372, un procedimiento de organización de una base de datos numéricos de manera que pueda hacerse un seguimiento. De manera más precisa, esta solicitud se refiere a un procedimiento de organización de una base de datos numéricos de manera que pueda hacerse un seguimiento, que comprende etapas de modificación de una base de datos numéricos principal mediante la adición o eliminación o modificación de un registro de la base principal y etapas de lectura de la base de datos principal, caracterizado porque la etapa de modificación de la base de datos principal comprende una operación de creación de al menos un registro numérico que comprende al menos:

los identificadores numéricos únicos de los registros y de los atributos en cuestión de la base de datos principal,

un identificador numérico único del estado de la base de datos principal correspondiente a dicha modificación de la base de datos principal,

los valores elementales de los atributos que se les han asignado a través de las operaciones elementales, sin proceder al almacenamiento de los atributos o de los registros no modificados,

y de adición de dicho registro en una base de historización interna compuesta por al menos una tabla,

y porque la etapa de lectura que se refiere a cualquier estado final o anterior de la base de datos principal consiste en recibir (o interceptar) una consulta original asociada al identificador único del estado previsto, en proceder a una transformación de dicha consulta original para construir una consulta modificada de direccionamiento de la base de historización que comprenda los criterios de la consulta original y el identificador del estado previsto, y de reconstrucción del o de los registros correspondientes a los criterios de la consulta original y al estado previsto, consistiendo dicha etapa de reconstitución en encontrar los valores elementales, contenidos en los registros de la base de historización, correspondientes a los criterios de la consulta original [con el fin de reducir las necesidades de capacidad de almacenamiento y los tiempos de tratamiento].

Se conoce también, a partir de la patente estadounidense US 6 292 795 (IBM), un sistema de indexación de archivos y un mecanismo para acceder a datos en un sistema de este tipo.

Finalmente, se conoce también en la técnica anterior, la patente estadounidense US 5 826 262 (IBM), un procedimiento de construcción en paralelo de árboles de radicales.

El problema técnico que la presente invención se propone resolver es el que consiste en la mejora de los rendimientos de las resoluciones de consultas en una base de datos. En efecto, los procedimientos de la técnica anterior consumen grandes recursos de máquina, tanto desde el punto de vista de los recursos de los procesadores como de los recursos de los discos.

Con este fin, la presente invención se refiere, en su acepción más general, a un procedimiento de organización de una base de datos relacional destinado a ponerse en práctica en una arquitectura informática que comprende al menos un procesador y memoria, caracterizado porque comprende las etapas que consisten en:

- elaborar una tabla de expansión jerárquica;
- crear un tesauro de cada una de las columnas;

- crear, para cada una de las palabras de cada uno de los tesauros, el árbol de los radicales del conjunto de los índices de filas en las que aparece dicha palabra;

- almacenar, para cada una de las claves primarias, la sucesión de estos valores poniendo en práctica una permutación sobre el conjunto de estos valores con el fin de encontrar un dato.

De manera ventajosa, el procedimiento comprende además una etapa de división de las tablas de la base de datos en un conjunto de subtablas, comprendiendo cada una un número dado de filas a excepción de la última subtabla. Preferiblemente, la base de datos pone en práctica el lenguaje SQL (*Structured Query Language*). La presente invención se refiere también a un sistema de base de datos organizado según el procedimiento de organización definido anteriormente. La presente invención se refiere también a un procedimiento de consulta de una base de datos organizada según el procedimiento de organización definido anteriormente, caracterizado porque comprende

- una primera etapa de cálculo de una tabla de expansión;

- una segunda etapa de resolución de la cláusula “Where” de la consulta interrogando las columnas de dicha tabla de expansión;

- una tercera etapa de interrogación de las imágenes no invertidas de las columnas para resolver la cláusula “Select”.

La invención se entenderá mejor con ayuda de la descripción, realizada a continuación a título meramente explicativo, de un modo de realización de la invención, en referencia a las figuras adjuntas:

- la figura 1 ilustra un almacenamiento por medio de un árbol de radicales;

- la figura 2 ilustra un ejemplo de representación de una columna de una tabla;

- la figura 3 ilustra un resumen del almacenamiento completo de una columna;

- las figuras 4 y 5 ilustran un árbol de radicales antes y después de una operación NOT.

Un árbol de radicales es un medio práctico de almacenar conjuntos de enteros, particularmente cuando están escritos en una misma longitud. Cuando se utilizan enteros, siempre es posible, por supuesto, imponerles la misma longitud de escritura (la del más largo o mayor) haciendo que su escritura vaya precedida del número adecuado de cifras 0.

Considérese, por ejemplo, un conjunto de enteros que se escriben en una longitud única en base 2, S = {0, 2, 5, 7, 11} = {0000, 0010, 0101, 0111, 1011}. Este conjunto puede almacenarse entonces en un árbol de radicales en el que cada camino, desde la raíz hasta las hojas, representa la escritura del entero almacenado en la hoja del árbol. Por ejemplo, el conjunto anterior puede almacenarse en el árbol de radicales de la figura 1. Las ventajas de utilizar un árbol de radicales son muy numerosas: el almacenamiento es económico en cuanto a espacio de memoria ya que los prefijos comunes a los distintos números sólo se almacenan una vez. Por otra parte, como se verá en los párrafos siguientes, las operaciones lógicas sobre los conjuntos almacenados de este modo son rápidas, económicas en cuanto a recursos de máquina y sencillas de implementar.

Se detalla cómo pueden los árboles de radicales ser útiles y eficaces para almacenar los datos de una base de datos o para modificarla.

En la primera parte, se examina el caso en el que la base de datos está compuesta por una única tabla, compuesta a su vez por una única columna.

A continuación, se examina el caso en el que la base de datos está compuesta por una única tabla, compuesta a su vez por varias columnas, y por al menos una clave primaria. En efecto, puede resultar muy práctico permitir que una tabla esté dotada de varias claves primarias. En efecto, en la práctica, sucede con frecuencia que una fila de tabla sólo esté rellena parcialmente. Puede suceder entonces que una de las claves primarias esté incompleta, y sea por tanto inutilizable, pero que otra esté completa.

Finalmente, la última parte se dedica a la creación de los índices de una base de datos cualquiera.

Una clave primaria es una columna, o un conjunto ordenado de columnas, de manera que dos filas diferentes de la tabla no puedan adoptar los mismos valores en esta (o estas) columna(s).

No obstante, siempre existe una clave primaria implícita y muy útil: el índice de la fila en la tabla (se trata, en efecto, de una clave primaria ya que dos filas distintas no pueden tener el mismo índice de fila). En lo sucesivo en la descripción de esta invención, se supondrá que esta clave primaria es efectiva.

Si ha de almacenarse, interrogarse y gestionarse una base de datos formada por una única tabla constituida a su vez por una única tabla, puede calcularse el tesauro de esta columna y, para cada palabra de este tesauro, calcular el conjunto de los índices de filas en los que aparece.

Estos índices de filas pueden almacenarse, naturalmente, en un árbol de radicales.

Ha de recalcarse que, en el transcurso de la creación del tesauro, se efectúa una ordenación de los datos. Se ordena, en efecto, el conjunto de los pares (palabra, índice de fila) según las palabras y, en caso de palabra igual, según los índices de filas. Así, por un lado puede calcularse el tesauro y, por otro lado, para cada una de las palabras de este tesauro construir el árbol de radicales de los índices de filas en los que aparece.

Tómese como ejemplo la tabla:

0	macho
1	hembra
2	hembra
3	macho
4	hembra
5	macho
6	macho
7	hembra
8	hembra
9	macho
10	macho

(en este ejemplo, los índices de filas se indican explícitamente).

Se construyen entonces los pares

(Macho, 0), (Hembra, 1), (Hembra, 2), (Macho, 3), (Hembra, 4), (Macho, 5), (Macho, 6), (Hembra, 7), (Hembra, 8), (Macho, 9), (Macho, 10) y se ordenan prioritariamente según su primer elemento: (Hembra, 1), (Hembra, 2), (Hembra, 4), (Hembra, 7), (Hembra, 8), (Macho, 0), (Macho, 3), (Macho, 5), (Macho, 6), (Macho, 9), (Macho, 10).

Puede construirse entonces el tesauro y, para cada palabra del tesauro, el conjunto de los índices de filas en los que aparece.

La palabra “Hembra” aparece en las filas {1, 2, 4, 7, 8} y “Macho” aparece en los índices {0, 3, 5, 6, 9, 10}.

Tras este trabajo, es muy fácil responder a una consulta del tipo ¿cuáles son los índices de filas en los que aparece la palabra “Macho”?

Estos conjuntos de índices de filas se almacenan en árboles de radicales. Este procedimiento de almacenamiento es muy útil para calcular la intersección, la unión, etc. de tales conjuntos.

En el ejemplo anterior, se obtiene el resultado presentado en la figura 2.

Otra consulta habitual es la que se refiere al contenido de una columna: la “entre” (*between*): es posible que se desee conocer los índices de filas cuyo contenido está comprendido entre dos límites dados.

Imagínese, por ejemplo, que una columna contiene datos, escritos en el formato AAAAMMDD. Comparar dos datos almacenados en este formato es en realidad lo mismo que compararlos lexicográficamente.

Sin embargo, también puede enriquecerse el tesauro con palabras obtenidas como truncamientos de las palabras del tesauro inicial. Por ejemplo, puede decidirse enriquecer el tesauro con todos los truncamientos de las cuatro o seis primeras letras de las palabras del tesauro inicial.

Así, cada palabra se representaría, en nuestro ejemplo, tres veces: una vez en tanto que ella misma, una vez truncada en seis caracteres y una última vez truncada en cuatro caracteres.

Cualquier palabra de seis caracteres, digamos aaaamm, aparecerá cada vez que la fila inicial contenga una palabra aaaammxx. En otras palabras, el conjunto de las filas en las que aparecerá la palabra aaaamm es la unión de los conjuntos de índices de filas en los que aparece una palabra de la forma aaaammxx (es decir, aaaamm seguido de lo que sea).

Igualmente, una palabra de cuatro caracteres aaaa aparecerá cada vez que esté presente una palabra de la forma aaaaxxyy en la tabla inicial. Su árbol de radicales es por tanto la unión de los árboles de radicales de las palabras de las que es prefijo.

El interés reside en que una cláusula “entre” (*between* en inglés) puede tratarse con una economía importante en cuanto a lecturas frente a un procedimiento de almacenamiento. Por ejemplo, si se busca el conjunto de las filas en las que aparece una fecha comprendida en el intervalo [19931117, 19950225], el número de lecturas necesarias de árboles de radicales es de 14+1+1+1+25 = 42 (ya que [19931117, 19950225] = [19931117, 19931130] U [199312, 199312] U [1994, 1994] U [199501, 199501] U [10050201, 19950225]), en lugar de 466.

Puede suceder en ocasiones que no se les haya proporcionado un valor a ciertas filas de una tabla. Sin embargo, para crear los árboles de radicales, toda fila debería tener un valor. Se eligen entonces valores que signifiquen que la fila correspondiente no contiene información. Naturalmente, se elegirá un valor que tenga relación con el tipo de datos almacenados; a modo de ejemplo, puede elegirse:

#Empty# para una cadena de caracteres, -2^{31} para un entero con signo de 32 bits, $2^{32} - 1$ para un entero sin signo de 32 bits, -2^{63} para un entero con signo de 64 bits, $2^{64} - 1$ para un entero sin signo de 64 bits, etc.

No obstante, el almacenamiento de una columna por tesauro y árboles de radicales no es muy práctico para responder a una consulta como “¿cuál es el valor de la fila 17?”, por ejemplo.

Es por esto que es necesario almacenar además la columna en su orden natural. Evidentemente, más que almacenar la columna en sí misma, será ventajoso a menudo almacenar la sucesión de los índices de palabras en el tesauro. Este almacenamiento complementario se denomina la imagen no invertida de la columna.

Por ejemplo, la columna anterior se almacenará de la manera siguiente:

Tesauro

0	Hembra
1	Macho

y la columna:

0
1
1
0
1
0
0
1
1
0
0

Nota: puede suceder que a medida que la base de datos se transforma, una palabra aparezca o desaparezca del tesauro (por ejemplo, cuando se retiran o se añaden filas a la tabla). Podría pensarse a partir de esto que es necesaria la reescritura completa de la columna. De hecho, éste no es el caso: más que almacenar el tesauro ordenado, puede almacenarse no ordenado y registrarse aparte una permutación que permita encontrar el orden lexicográfico de las

palabras que lo componen. Es por ello que, si una palabra aparece en el tesauro, no es necesaria la reescritura completa de la columna. En este caso se reescribe la permutación que permite encontrar el orden lexicográfico de las palabras más que el propio tesauro. La figura 3 ilustra el resumen del almacenamiento completo de una columna.

En el caso de una base de datos que sólo contenga una única tabla constituida a su vez por varias columnas, puede tratarse como si estuviese formada por columnas independientes. En otras palabras, puede crearse el almacenamiento de cada una de las columnas que constituyen la tabla.

La única cuestión pendiente es entonces el tratamiento de la clave primaria.

Cuando se trata de una clave primaria, es necesario que se pueda responder de la manera más rápida posible a dos tipos de consultas opuestas: "¿En qué fila puede encontrarse un valor dado de clave primaria?" y "¿Cuál es el valor de la clave primaria encontrada en una fila dada?".

Puede responderse de manera eficaz a estos dos tipos de interrogación almacenando a la vez la columna o las columnas que forman la clave primaria en el orden en el que las filas aparecen en la tabla y una permutación que permita leer las columnas en el orden asociado a una función cualquiera de comparación. Puede encontrarse por tanto un valor dado mediante dicotomía.

Por ejemplo, imagínese que una clave primaria está formada por dos columnas, cuyos valores se almacenan en la tabla siguiente.

(0)	1	3
(1)	2	1
(2)	3	2
(3)	2	3
(4)	1	2
(5)	3	7
(6)	2	2
(7)	1	1
(8)	3	3
(9)	4	3

En este ejemplo, los índices de filas se expresan de nuevo de manera explícita, pero puestos entre paréntesis. Se almacenan por tanto las dos columnas exactamente como están en la tabla y una permutación, asociada a una función de comparación elegida. Por ejemplo, puede decidirse comparar en primer lugar la primera columna lexicográficamente y, en caso de valor igual, comparar la segunda de manera ordinal.

En este caso, la clave primaria ordenada es:

(7)	1	1
(4)	1	2
(0)	1	3
(1)	2	1
(6)	2	2
(3)	2	3
(2)	3	2
(8)	3	3
(5)	3	7
(9)	4	3

Retirando los valores (pero conservando los índices), se obtiene la permutación (7401632859). El valor más pequeño se encuentra por tanto en la fila 7, el segundo más pequeño en la fila 4, etc. Encontrar un valor dado se realiza así fácilmente mediante dicotomía.

Cuando se almacena una tabla, es muy práctico almacenar y mantener actualizado el número total de filas de la que está constituida.

En una base de datos relacional, habitualmente hay varias tablas conectadas entre sí mediante juegos de claves primarias, claves externas.

Como se ha explicado anteriormente, una clave primaria es una columna o un conjunto ordenado de columnas que no pueden adoptar el o los mismos valores en dos filas distintas. (El índice de fila es un ejemplo fundamental de clave primaria).

Supóngase que una tabla esté constituida por varios millones de filas, pero que algunas de sus columnas sólo pueden adoptar un número muy reducido de valores diferentes (por ejemplo, una base de datos que contenga datos de genealogía puede contener los nombres de personas, para cada una de ellas su país de nacimiento, su continente de nacimiento, los países y continentes de nacimiento de su madre y de su primer hijo si tiene). En lugar de proporcionar un valor a todas estas columnas, se considera muy económico almacenar, en tal caso, los países en una tabla separada de la tabla principal y los continentes en una tercera tabla. La tabla principal contiene entonces, en cada fila, un valor (una clave externa) que proporciona un identificador de fila (un valor de clave primaria) de la tabla “países”; y la tabla “países” contiene, en cada una de sus filas, un valor (una clave externa) que identifica una de las filas de la tabla “continentes” (clave primaria).

He aquí un ejemplo (tabla de clientes a continuación) muy escueto, que ilustra lo anterior:

(fi)	Cn	Inc	Bircoun	BirCont	MoCoun	MoCont	EldCoun	BldCont
(0)	Dupont	817	Francia	Europa	Túnez	África	Inglaterra	Europa
(1)	Gracamoto	1080	Japón	Asia	Japón	Asia	EE.UU.	América
(2)	Smith	934	Inglaterra	Europa	India	Asia	Inglaterra	Europa
(3)	Helmut	980	Alemania	Europa	Alemania	Europa	Alemania	Europa

(en este ejemplo, “cn” designa el nombre, “inc” los ingresos, “Bircoun” el país de nacimiento, “BirCont” el continente de nacimiento, “MoCoun” el país de nacimiento de la madre, “MoCont” el continente de nacimiento de la madre, “EldCoun” el país de nacimiento del primogénito y “EldCont” el continente de nacimiento del primogénito.

Esta tabla puede reescribirse en varias tablas:

Continentes:

Fi)	Continente
0)	África
1)	América
2)	Asia
3)	Europa

Países:

fi)	País	Continente
0)	Francia	3
1)	Túnez	0
2)	Inglaterra	3
3)	Japón	2
4)	EE.UU.	1
5)	India	2
6)	Alemania	3

La tabla principal queda de la siguiente manera:

(fi)	en	Inc	Bircoun	MoCoun	EldCoun
(0)	Boyer	817	0	1	2
(1)	Gracamoto	1080	3	3	4
(2)	Smith	934	2	5	2
(3)	Belmut	980	6	6	6

El conjunto de las tres tablas así construidas, ocupa ciertamente menos espacio que la tabla inicial.

Sin embargo, esto ilustra también la idea según la cual una base relacional puede transformarse en un conjunto de tablas independientes unas de otras.

En el ejemplo anterior, puede considerarse la tabla “continentes” por sí sola, la tabla “países” con la tabla “continentes” desarrollada dentro (es decir, la tabla “países” en la cual las referencias a la tabla “continentes” se han sustituido por las filas de la propia tabla) y la tabla “personas” con las tablas “países” y “continentes” desarrolladas dentro.

Las tablas de expansión son entonces:

Tabla de expansión Continentes:

(fi)	Continente
(0)	África
(1)	América
(2)	Asia
(3)	Europa

La tabla de expansión Países queda como:

fi)	País	Continente
0)	Francia	Europa
1)	Túnez	África
2)	Inglaterra	Europa
3)	Japón	Asia
4)	EE.UU.	América
5)	India	Asia
6)	Alemania	Europa

Tabla de expansión Clientes:

(fi)	Cn	Inc	Bircoun	BirCont	MoCoun	MoCont	EldCoun	EldCont
(0)	Boyer	817	Francia	Europa	Túnez	África	Inglaterra	Europa
(1)	Gracamoto	1080	Japón	Asia	Japón	Asia	EE.UU.	América
(2)	Smith	934	Inglaterra	Europa	India	Asia	Inglaterra	Europa
(3)	Helmut	980	Alemania	Europa	Alemania	Europa	Alemania	Europa

Evidentemente puede suceder, como en este ejemplo, que una tabla deba desarrollarse varias veces en otra. Esto implica que una columna de tabla desarrollada en otra siempre deberá referenciarse como perteneciente a la tabla de expansión, desarrollada en la tabla de expansión a través de un juego de claves primarias y claves externas que formarán parte de la identidad de la columna.

Una tabla de expansión se define por tanto como una tabla en la que todas las tablas que puedan desarrollarse en ella, lo están, en tantos ejemplares como juegos de claves primarias y claves externas hay, que permiten pasar de la tabla de expansión a la tabla desarrollada. Una vez efectuado este procedimiento de expansión, la base de datos está formada por tablas de expansión independientes unas de otras.

Para cada una de estas tablas de expansión, se construyen los índices tal como se ha explicado en el caso de una tabla única.

Tras la etapa de expansión y la de indexación, la invención descrita en el presente documento se refiere también a un método de tratamiento de consultas y modificación de la base de datos estructurada e indexada de este modo.

En esta parte, se explicará cómo los índices creados se utilizan para resolver de manera eficaz consultas SQL. Habitualmente, una consulta implica varias tablas y puede separarse en dos partes distintas: la cláusula “where” que solicita al gestor de bases de datos que calcule índices de filas de una tabla y una parte que solicita al gestor de bases de datos que realice cálculos sobre los datos que se encuentran en las filas calculadas.

La primera parte puede contener uniones de tablas (un enlace entre una clave externa y la clave primaria correspondiente), una comparación entre una columna y una constante (con un conector aritmético como =, >=, >, <, <=, *Between, like, in...*) o una comparación entre dos columnas (mismos operadores aritméticos o incluso un producto cartesiano). Estas consultas están conectadas entre sí, cuando hay varias, mediante conectores lógicos (*and, or, not...*).

La segunda parte de la consulta puede contener operaciones aritméticas como sumas, medias, productos, el operador de recuento*, etc.

Como se ha explicado anteriormente, cada una de las tablas se considera como tabla de expansión, lo que significa que las uniones de tablas no son pertinentes para una tabla de este tipo.

Sin embargo, una consulta comprende habitualmente varias tablas. Se plantea por tanto el problema de la elección de la tabla de expansión en la que resolver la consulta, que el procedimiento resuelve de la siguiente manera.

Las tablas implicadas en la consulta se desarrollan todas en un conjunto no vacío, sea T. Una sola de estas tablas de expansión no está desarrollada en las otras. Esta tabla es la tabla en la que debe resolverse la consulta.

La cláusula "where" contiene por tanto cláusulas de uniones, conectadas de manera lógica al resto de la consulta mediante conexiones lógicas "y". Por tanto basta con simplemente borrarlas sustituyendo toda la cláusula "y" por el otro término. Esto significa que se sustituye "(unión Y resto)" por "Resto" para cada cláusula de unión.

Véase ahora cómo la invención gestiona de manera eficaz una cláusula "where" desprovista de sus cláusulas de unión.

Se denomina "consulta atómica" una parte indivisible de la cláusula *where*, es decir, una comparación que forma toda la consulta o que está conectada al resto de la consulta por conectores lógicos sin que ella misma los contenga. Si una tabla t comprende la columna c, una consulta atómica será, por ejemplo, t.c = 3, t.c between "HIGH" and "MEDIUM", o incluso t.c like Word%.

Los siguientes párrafos explican cómo la invención gestiona las consultas atómicas.

El caso más sencillo de tratar es aquél en el que hay igualdad entre una columna y un valor dado. Se lee el árbol de radicales del valor buscado para la columna de la consulta.

La cláusula "Between" es un ejemplo fundamental de consulta atómica. Todas las demás consultas atómicas pueden remitirse a este caso. Es para esta cláusula para la que se han creado las macro-palabras.

Retómese el ejemplo de las fechas dado en el párrafo sobre las macro-palabras. Esta columna se ha gestionado enriqueciendo el vocabulario con los truncamientos de sus palabras de longitud 4 y de longitud 6. Si se buscan las filas cuyo valor está comprendido entre [19931117, 19950225], se divide el intervalo en: [19931117, 19950225] = [19931117, 19931130] U [199312, 199312] U [1994, 1994] U [199501, 199501] U [10050201, 19950225].

El cálculo es entonces muy sencillo: se lee el árbol de radicales del valor 19931117, que se une (operación lógica "or") con el del valor 19931118,... que se une con el del valor 19931130 y a continuación con el del valor (truncado en 6 caracteres) de 199312 y después con el (truncado en 4 caracteres) de 1994 y a continuación con el (truncado en 6 caracteres) de 199501, y después con el de 19950201 y a continuación con el de... 19950225.

Así se leen 42 árboles de radicales en lugar de los 466 que habría hecho falta leer sin las macro-palabras.

El tratamiento de "or" se explica más adelante.

Por supuesto, pueden tratarse intervalos semiabiertos o abiertos excluyendo simplemente las palabras correspondientes.

Cada una de las consultas atómicas "mayor o igual, menor o igual, mayor, menor" es una cláusula *between* que se ignora. En efecto, si se denomina m y M el mínimo y el máximo del tesauro de la columna en cuestión, entonces pueden tratarse estas cláusulas como una cláusula *between*.

t.c > a	Significa que t.c está dentro de	]a,M]
t.c >= a	Significa que t.c está dentro de	[a,M]
t.c < a	Significa que t.c está dentro de	[m,a[
t.c <= a	Significa que t.c está dentro de	[m,a]

La cláusula *In* es una forma de mezclar igualdades mediante "or". Se gestionan de manera muy sencilla.

Por ejemplo, t.c in (a,b,c) se reescribe en t.c = a or t.c = b or t.c = c. La gestión de las cláusulas "or" se explica más adelante.

La cláusula “like” es otro ejemplo de cláusula *between*. Por ejemplo, la cláusula t.c like Mot% se reescribe en efecto en t.c between [Mot, Mou].

Las consultas atómicas pueden mezclarse con ayuda de conectores lógicos: “And”, “Or” y “Not”. Las tres subsecciones siguientes están dedicadas a estos operadores.

En primer lugar se desea insistir en el hecho de que una consulta atómica devuelve siempre en el árbol de radicales, lo que será también el caso de estos operadores lógicos y por último de la cláusula *where*.

El hecho de realizar “or” (“o”) en dos árboles de radicales es de hecho la operación que consiste en unirlos. Este cálculo se realiza muy fácilmente mediante un recorrido a lo ancho de los dos árboles simultáneamente.

Se realiza de manera recursiva mediante

unión (t1, t2)

Inicio

Árbol res;

Si (t1 = NULL) res = t2

Si (t2 = NULL) res = t1

res-> HijoIzquierdo = Unión(t1-> HijoIzquierdo, t2->HijoDerecho)

res-> HijoDerecho = Unión (t1-> HijoDerecho, t2-> HijoDerecho)

Devolver res

Fin

La cláusula “and” se calcula casi como la anterior (corresponde a una intersección):

Intersección (t1, t2)

Inicio

Árbol res;

Si (t1 = NULL) res = NULL

Si (t2 = NULL) res = NULL

res->HijoIzquierdo = Intersección (t1-> HijoIzquierdo, t2-> HijoIzquierdo)

res-> HijoDerecho = Intersección (t1-> HijoDerecho, t2-> HijoDerecho)

devolver res

Fin

Esta cláusula requiere sin embargo menos tiempo de cálculo de media que la anterior. En efecto, durante los dos recorridos de los árboles en paralelo, basta con que uno de los dos nodos explorados no tenga hijo izquierdo para que la exploración del hijo izquierdo del otro nodo sea innecesaria.

Esto es así, particularmente, cuando los árboles se han almacenado en un disco duro en archivos separados.

La cláusula “Not” es una de las más difíciles de poner en práctica de entre las consultas atómicas.

El índice máximo de las filas de cada una de las tablas se almacena y actualiza. La cláusula *Not* se trata entonces como sigue (el objetivo es calcular el árbol de radicales *Not* T, siendo T un árbol de radicales).

Se define en este caso un árbol n de radicales completo, un árbol de radicales que contiene todos los valores de enteros comprendidos entre 0 y n-1.

Para calcular "not", se retiran entonces de manera recursiva, con ayuda de x-*or* las hojas de T de un árbol n de radicales (donde n designa el índice máximo de las filas de la tabla de expansión a la que pertenece T).

Cuando se retira un nodo de un árbol de radicales, se retira éste y se retira de manera recursiva su padre si no tiene más hijos.

Por ejemplo, la figura 3 muestra el cálculo de *Not* T cuando la tabla de expansión a la que pertenece T tiene un índice máximo de filas utilizado igual a 13.

El árbol inicial se presenta en la figura 3 y el árbol transformado se presenta en la figura 4.

La comparación entre dos columnas es la más compleja de las consultas atómicas. Esta consulta se trata prácticamente como un producto cartesiano (véase la sección siguiente).

Considérese una tabla de expansión t y dos de sus columnas t.c y t.d. Una comparación entre estas dos columnas es una operación en el curso de la cual se desea discriminar las filas de t para las que t.c > t.d por ejemplo. Se recalca el hecho de que esta comparación se realiza en caso de índice de fila idéntico para las dos columnas, lo que distingue esta comparación del producto cartesiano.

¿Cómo se realiza esta consulta? Sean T_c y T_d los tesauros de las dos columnas t.c y t.d.

Se buscan aquellas filas en las que t.c > t.d. He aquí cómo proceder. Para cualquier palabra *w* del tesauro T_c, se calcula el árbol de radicales *r* del intervalo [m_d, w'] donde *w'* designa la palabra más grande de T_d estrictamente menor que *w*. Entonces, al calcular "y" sobre *r* y el árbol de radicales de *w*, se obtiene un árbol t_w.

Al efectuar la unión de todos los árboles de radicales r_w, se obtiene el árbol de radicales buscado.

Es evidente que los árboles t_w no tienen que calcularse de manera independiente unos de otros. Dado que las palabras *w* se recorren en orden creciente, basta con unir t_w en el árbol correspondiente añadiendo las palabras comprendidas entre *w* y la palabra siguiente en T_c.

También puede calcularse t_c-t_d con ayuda de archivos planos y leer el resultado.

Las otras cláusulas parecidas se resuelven de manera similar (por ejemplo t.c >= t_d).

Examínese el tratamiento de las subconsultas. En efecto, puede suceder que una cláusula "where" contenga a su vez otra cláusula "where", que puede estar o no correlacionada con la cláusula "where" principal.

¿Qué es una subconsulta correlacionada? Un ejemplo de una consulta de este tipo se proporciona mediante la consulta 17 del TPC. Esta consulta es:

```
select
sum(l_extendedprice) / 7.0 as avg_yearly
from
 lineitem
 part
where
 p_partkey = l_partkey
 and p_brand = ' [BRAND]'
 and p_container = '[CONTAINER]'
 and l_quantity < (
   select
   0.2 * avg(l_quantity)
  from
   lineitem
  where
   p_partkey = p_partkey
```

En esta consulta, debe realizarse el cálculo de una subconsulta teniendo en cuenta las condiciones requeridas por la consulta principal (ya que *p_partkey* de la subconsulta pertenece a la cláusula *where* principal).

Por tanto, este tipo de consulta puede reescribirse de modo que se cambie esta subconsulta por una subconsulta no correlacionada. Para ello basta con duplicar las condiciones requeridas por la cláusula *where* principal en la cláusula *where* de la subconsulta correlacionada. En nuestro ejemplo, esto da como resultado:

```
select
   sum(l_extendedprice)/ 7.0 as ag_yearly
from
  lineitem
  part
where
  p_partkey = l_partkey
  and p_brand = '[BRAND]'
  and p_container = '[CONTAINER]'
  and l_quantity < (
    select
     0.2 * avg(l_quantity)
    from
     lineitem
     partsupp
   where
     p_part<key = p_partkey
     and p_brand = '[BRAND]'
     and p_container = '[CONTAINER]'
  );
}
```

Finalmente, una subconsulta correlacionada puede reescribirse en una subconsulta no correlacionada.

Se trata una consulta SQL que contiene subconsultas no correlacionadas tratando en primer lugar las subconsultas de manera recursiva y sustituyendo en la consulta cada subconsulta por su resultado.

Así, una cláusula “where” que devuelve un árbol de radicales se trata representando los índices de filas de la tabla de expansión correspondientes a esta cláusula.

Supóngase ahora que el objeto de la consulta sea realizar ciertos cálculos sobre algunas columnas en las filas encontradas. Por ejemplo, es posible que quiera calcularse la media en las filas encontradas de los valores de una determinada columna.

Los valores de esta columna están almacenados “planos” en su orden de aparición. Por tanto es muy sencillo releer los valores de esta columna únicamente para las filas encontradas anteriormente y efectuar sobre estos valores los cálculos solicitados.

La invención se ha descrito en lo que antecede a modo de ejemplo. Se entiende que el experto en la técnica podrá realizar diferentes variantes de la invención sin salirse por ello del marco de la patente.

REIVINDICACIONES

1. Procedimiento para la aceleración de la interrogación de una base de datos, de organización de una base de datos relacional y para la reducción del espacio de almacenamiento requerido, destinado a ponerse en práctica en una arquitectura informática que comprende al menos un procesador y memoria, que comprende las etapas que consisten en:

- registrar, para cada una de las tablas de la base de datos relacional, una tabla de expansión jerárquica constituida por una tabla en la que se han desarrollado todas las tablas que podían desarrollarse en la misma,

- calcular los índices para esta tabla de expansión, poniendo en práctica las etapas siguientes:

  - crear un tesauro de cada una de las columnas de dicha tabla de expansión jerárquica;

  - crear, para cada una de las palabras de cada uno de los tesauros, el árbol de los radicales del conjunto de los índices de filas en las que aparece dicha palabra;

caracterizado porque cada árbol de cada una de las palabras es un árbol de radicales y porque el procedimiento comprende, durante la recepción de una consulta de interrogación:

- una primera etapa de determinación de la tabla de expansión;

- una segunda etapa de resolución de la cláusula "Where" de la consulta interrogando las columnas de dicha tabla de expansión, efectuándose esta resolución de la cláusula *where* resolviendo las comparaciones entre columnas mediante lectura de los árboles de radicales de las palabras correspondientes, y a continuación realizando en los árboles de radicales anteriormente mencionados las operaciones de intersección, de unión o de complementación correspondientes a las operaciones lógicas (*and*, *or*, *not*) de la cláusula *where*.

2. Procedimiento de organización de una base de datos según la reivindicación 1, **caracterizado** porque comprende además una etapa que consiste en almacenar, para cada una de las claves primarias, la sucesión de sus valores poniendo en práctica una permutación sobre el conjunto de estos valores con el fin de encontrar un dato.

3. Procedimiento de organización de una base de datos relacional según la reivindicación 1, **caracterizado** porque comprende además una etapa de interrogación de las imágenes no invertidas de las columnas para resolver la cláusula "Select".

4. Procedimiento de organización de una base de datos según la reivindicación 1 ó 2 ó 3, **caracterizado** porque comprende además una etapa de dividir las tablas de la base de datos en un conjunto de subtablas, comprendiendo cada una un número dado de filas a excepción de la última subtabla.

5. Procedimiento de organización de una base de datos según la reivindicación 1 ó 2 ó 3 ó 4, **caracterizado** porque la base de datos pone en práctica el lenguaje SQL (*Structured Query Language*).

6. Sistema de base de datos organizado según el procedimiento de organización de la reivindicación 1, 2 ó 3 ó 4 ó 5.

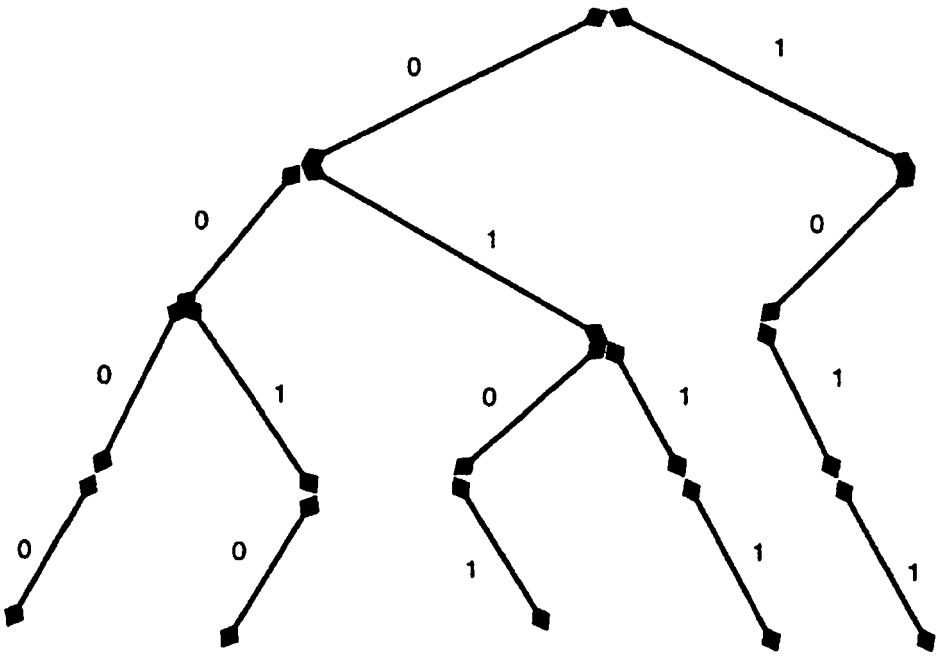
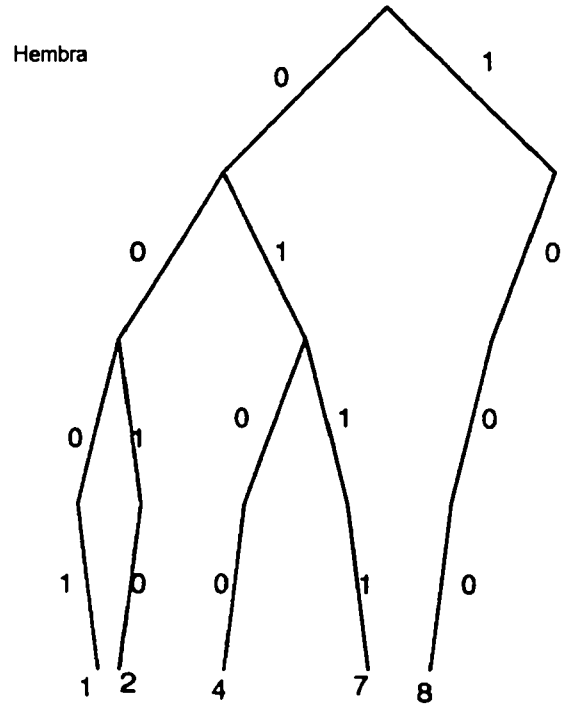


Figura 1


Hembra
0
1
0
1
0
0
1
0
1
0
1
0
1
0
1
0
1
1
2
4
7
8

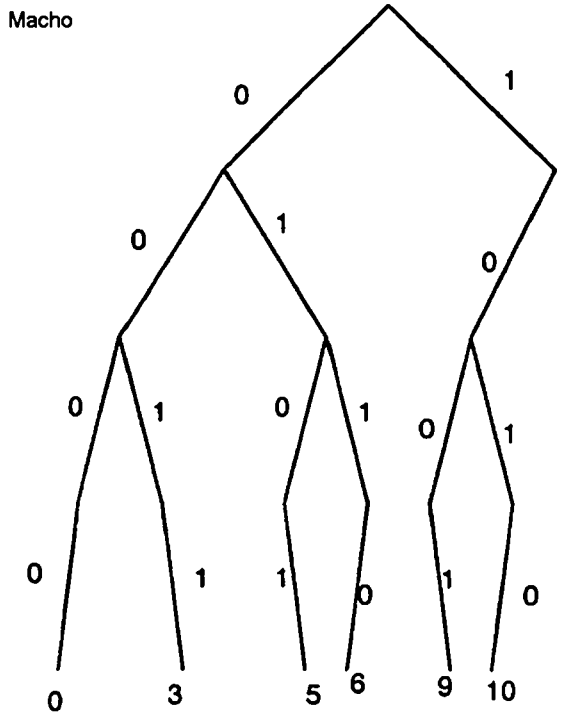

Macho
0
1
0
1
0
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
3
5
6
9
10

Figura 2

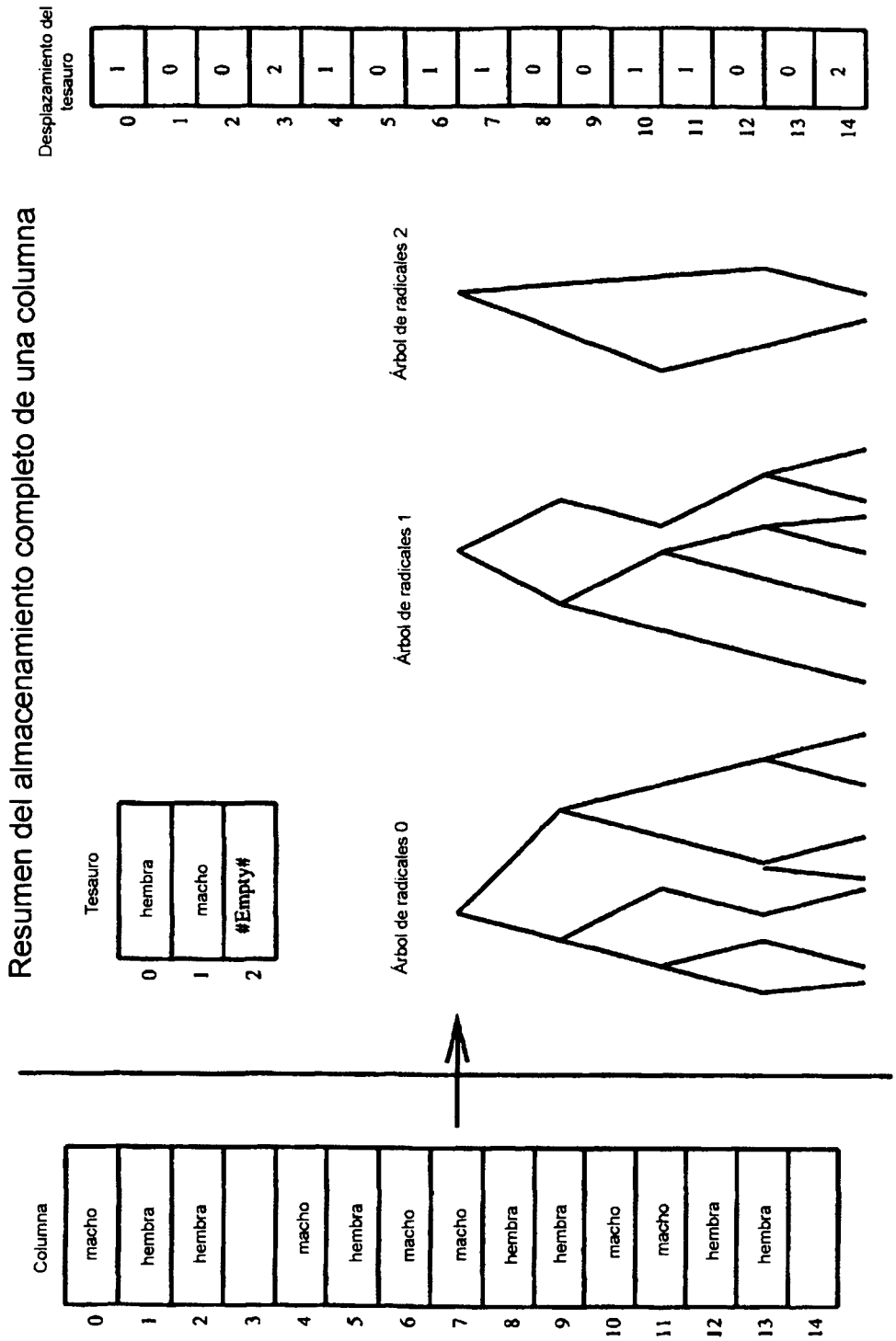

Resumen del almacenamiento completo de una columna
Columna
0 macho
1 hembra
2 hembra
3
4 macho
5 hembra
6 macho
7 macho
8 hembra
9 hembra
10 macho
11 macho
12 hembra
13 hembra
14
Tesauro
0 hembra
1 macho
2 #Empty#
Árbol de radicales 0
Árbol de radicales 1
Árbol de radicales 2
Desplazamiento del tesauro
0 1
1 0
2 0
3 2
4 1
5 0
6 1
7 1
8 0
9 0
10 1
11 1
12 0
13 0
14 2

Figura 3

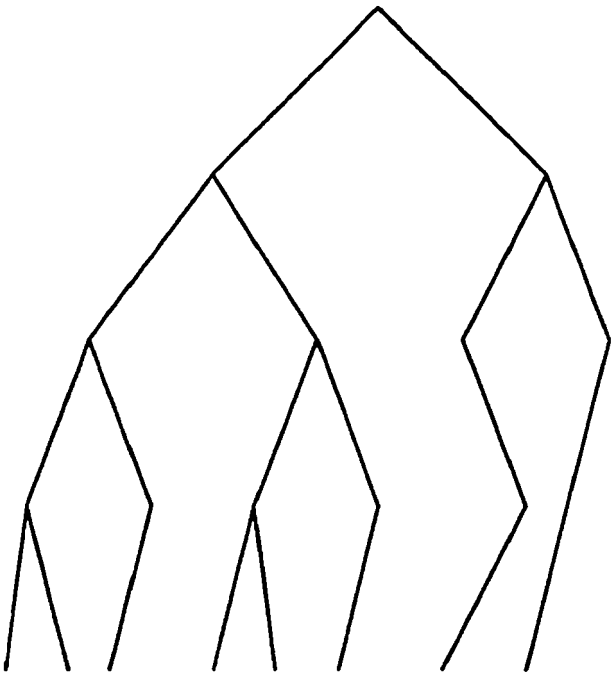


Figura 4

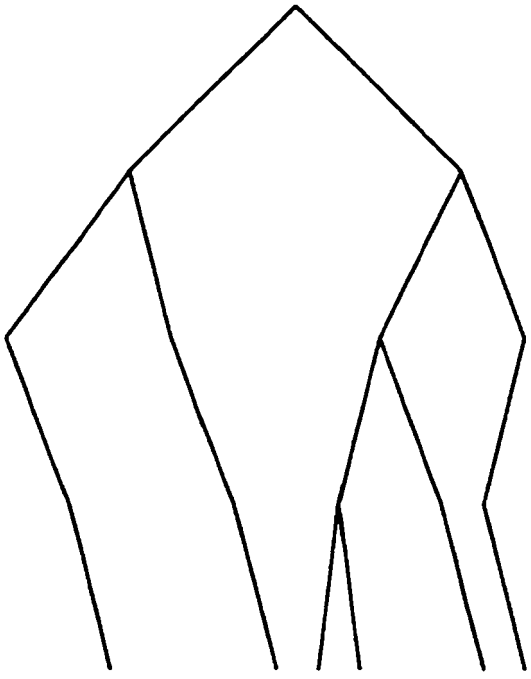


Figura 5