(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 636 758**

(51) Int. CI.:

G06F 17/30 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **30.08.2013 E 13461545 (9)**

(97) Fecha y número de publicación de la concesión europea: **10.05.2017 EP 2843567**

(54) Título: **Procedimiento implementado por ordenador para mejorar la ejecución de consulta en bases de datos relacionales normalizadas en el nivel 4 y superior**

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
09.10.2017

(73) Titular/es:

PILAB S.A. (100.0%)
ul. Rzeznicza 32-33
50-130 Wroclaw, PL

(72) Inventor/es:

PIECKO, KRYSTIAN

(74) Agente/Representante:

CARPINTERO LÓPEZ, Mario

DESCRIPCIÓN

Procedimiento implementado por ordenador para mejorar la ejecución de consulta en bases de datos relacionales normalizadas en el nivel 4 y superior

El objeto de la presente invención es un procedimiento implementado por ordenador para optimización de la ejecución de consulta de bases de datos. El procedimiento encuentra su aplicación en sistemas de recuperación y procesamiento de datos.

De acuerdo con WIKIPEDIA (www.wikipedia.org), las consultas de tipo JOIN se definen en la técnica anterior como sigue. Una clausula JOIN de SQL (Lenguaje de Consulta Estructurado, del inglés Structured Query Language) combina registros de dos o más tablas en una base de datos. Crea un conjunto de datos que se puede guardar como una tabla o usarse tal cual. Una consulta JOIN es un medio para combinar cambios de dos o más tablas mediante el uso de valores comunes en cada una. El estándar ANSI SQL especifica cuatro tipos de JOIN: INNER, OUTER, LEFT, y RIGHT. Como caso especial, una tabla (tabla base, vista, o una tabla unida) puede UNIRSE a sí misma en una autounión.

Un programador escribe un predicado JOIN para identificar los registros que unirá. Si el predicado evaluado es verdadero, el registro combinado se produce, entonces, en el formato esperado, un registro establecido o tabla temporal.

Las bases de datos relacionales actuales a menudo se normalizan con el fin de eliminar información duplicada cuando los objetos pueden tener relaciones uno a uno, uno a muchos o muchos a muchos. Por ejemplo, se puede asociar un único Departamento asociado con muchos Empleados diferentes. Unir dos tablas de manera eficaz crea otra tabla que combina información de ambas tablas. Esto es a un cierto coste en términos del tiempo que toma para computar la unión.

Esto es a un cierto coste en términos del tiempo que toma para computar la unión, ya que los sistemas relacionales muy comúnmente llaman a unirse, aunque se enfrentan a dificultades para optimizar su ejecución eficiente. En casos en los que una consulta requiere unir más de una estructura de datos, el tiempo de ejecución puede aumentar exponencialmente.

Existen tres algoritmos para realizar operaciones de tipo JOIN: una unión de bucle anidado que s un algoritmo s imple que une dos conjuntos usando dos bucles anidados; una unión de clase combinación que es un algoritmo para ordenar primero las r elaciones por el atributo de unión, de modo que las exploraciones lineales entrelazadas encontrarán estos conjuntos al mismo tiempo; y una unión hash que es un algoritmo realizado por la ejecución de un algoritmo hash de u n conjunto de datos en una memoria basada en la unión de columnas y la lectura del otro y probar la tabla hash en busca de coincidencias.

Como se mencionó previamente, las bases de datos relacionales a menudo se normalizan. El objetivo de la normalización es almacenar solo la mínima cantidad de información, para eliminar redundancias en los datos, para eliminar anomalías y para reestructurar datos con el fin de proporcionar almacenamiento más eficiente.

El concepto de normalización de datos se desarrolló por E.F. en 1970. El Sr. Codd en su publicación sobre "*Further Normalization of the Data Base Relational Model*" definió formas normales para reducir la cantidad de redundancia y dependencia de inconsistencia dentro de las bases de datos. El Sr. Codd definió tres formas normales, pero durante los años posteriores dos formas normales más se introdujeron.

Existen cuatro formas normales más comúnmente usadas: la forma normal primera (1NF), la segunda (2NF) y la tercera (3NF), y a veces se usa la forma Boyce-Coddnormal (BCNF).

También se conoce una cuarta forma (4NF) normal, que es una forma normal usada en la normalización de la base de datos. Introducido por Ronald Fagin en 1977, 4NF es el siguiente nivel de normalización después de la forma normal de Boyce-Codd (BCNF). Considerando que la segunda, la tercera, y la forma normal Boyce-Codd se ocupan de las dependencias funcionales, 4NF se refiere a un tipo más general de dependencia conocida como una dependencia conocida como una dependencia de valor múltiple [fuente: WIKIPEDIA].

También existe en la técnica anterior, una Quinta forma (5NF) normal, también conocida como forma (JPNF) normal de unión-proyección, que establece que no existe ninguna dependencia de unión no trivial. El 5NF establece que cualquier hecho debería poder reconstruirse sin ningún resultado anómalo en ningún caso, independientemente del número de tablas que se están uniendo. Una tabla 5NF debería tener solo claves candidatas y su clave primaria debería consistir en solo una única columna.

El problema con estas formas normales es el tamaño de las uniones que se requieren para reconstruir cualquier dato no trivial. Un desarrollador puede depender de vistas y procedimientos para simplificarlos, pero los datos subyacentes todavía terminan siendo muy complejos. También existen problemas de rendimiento a considerar - que es por lo que 4NF y 5NF son a menudo académicos. En la mayoría de los casos, 3NF (o BCNF) se implementan. (fuente: http://www.devshed.com/c/a/Administration/Database- Normalization/4/).

La normalización produce muchas tablas, que tiene cada una relativamente pocas columnas, por ejemplo, dos columnas - una clave primaria y un valor. Para usar los datos de normalización contenidos en estas columnas, uno tiene que poner la información de vuelta junta uniendo las columnas usando sus relaciones de claves primarias/remotas.

Ejecutar consultas a una base de datos normalizada normalmente requiere recuperación de datos almacenados para múltiples tablas normalizadas. La base de datos normalizada, por lo tanto, necesita localizar y recuperar las tablas solicitadas y, entonces, une la información de las bases de datos para responder a la solicitud de datos.

Las solicitudes de tipo JOIN reducen el rendimiento de la base de datos ralentizando el proceso y colocando tensión de procesamiento pesado sobre el hardware informático. Las bases de datos normalizadas necesitan mayor CPU, memoria, e I/O para procesar transacciones y consultas que las bases de datos no normalizadas y desnormalizadas. En las bases de datos existentes, las consultas de tipo JOIN incurren en sobrecarga de procesamiento significante, que conduce a la incapacidad para trabajar en tiempo real.

Por lo tanto, podría ser muy ventajoso aumentar el rendimiento de las consultas de la base de datos, especialmente en las consultas de tipo JOIN ya que son cruciales para las bases de datos normalizadas.

Algunas bases de datos de la técnica anterior implementan las llamadas técnicas de optimización de consulta. La optimización de la consulta es una función de sistemas de gestión de bases de datos, tales como sistemas (RDBMS) de gestión de bases de datos relacionales. El optimizador de consultas trata de determinar el modo más eficiente de ejecutar una consulta dada considerando los planes de consulta posibles. En implementaciones típicas, el optimizador de consulta no puede ser accesible directamente por los usuarios RDBMS: una vez que las consultas se envían a un servidor de base de datos, y se analizan por el analizador, pasan seguidamente al optimizador de consulta donde tiene lugar la optimización.

Por ejemplo, una publicación de patente US 5.548.758 titulada "*Optimization of SQL queries using early-out join transformations of column-bound relational tables*" desvela un procedimiento y aparato para optimizar solicitudes SQL en un sistema de gestión de bases de datos relacionales que usa transformaciones de unión de retiro anticipado. Una unión de retiro anticipado comprende una unión existencia de muchos a uno, en la que la unión busca una coincidencia en una tabla interior para cada fila de la tabla externa y termina la búsqueda para cada fila de la tabla externa cuando una única coincidencia se encuentra en la tabla. Para transformar una unión de muchos a muchos para una unión de retiro anticipado, la consulta debe incluir un requisito de carácter distintivo, tanto explícita como implícitamente, en una o más columnas de resultado para cada operación de unión. El carácter distintivo puede especificarse usando la palabra clave DSTINCT en la cláusula SELECT o puede ser implícita a partir de los predicados presentes en la consulta. La transformación de unión de retiro anticipado también necesita que no se haga referencia a ninguna columna en la tabla interna después de la unión, o si se hace referencia a una columna de tabla interna después de la unión, de que cada columna referenciada sea "obligada". Una columna referenciada puede ser obligada de una de tres maneras: (1) una columna de tabla interna puede ser obligada a una constante a través de un predicado de igualdad, (2) una columna de tabla interna puede ser obligada a una columna de tabla externa, o (3) una columna de tabla interna puede ser obligada a un valor correlacionado, en el que el valor correlacionado origina el bloqueo de consulta. En los tres casos, una columna de tabla interior puede ser obligada a través de transitividad de predicados de igualdad.

Una solicitud de patente US20120246147 titulada "*MODULAR QUERY OPTIMIZER*", desvela programas informáticos codificados en un medio de almacenamiento informático que proporciona un optimizador de consulta modular. En un aspecto, un producto programa informático incluye seleccionar una o más proyecciones de un conjunto de proyecciones para cada tabla en una consulta de base de datos en la que cada una de las proyecciones seleccionadas para la tabla ha llevado a costes de ejecución inferiores estimados para la consulta en comparación con las proyecciones no seleccionadas; generar órdenes de unión para la consulta basada en distribución de datos de una o más de las proyecciones seleccionadas entre sitios en una red informática en la que las órdenes de unión reflejan diferentes combinaciones de operaciones de distribución de datos aplicadas a la salida de una o más uniones de consulta; y seleccionar una orden de unión desde las órdenes de unión basándose en las órdenes de unión que usan un modelo de coste.

Los inconvenientes de la optimización de consulta conocidos en las bases de datos normalizadas incluyen, por ejemplo, requisitos de mayor memoria y CPU y dificultades en formular consultas complejas.

Hasta ahora, tales problemas se han dirigido con un uso de un hardware más poderoso, tal como servidores de bases de datos que tienen mayor rendimiento y más memoria en lugar de soluciones relacionadas con el diseño de bases de datos y de ejecución de consultas.

Por ejemplo, una solicitud de patente US2012117027 titulada "Methods and systems for hardware acceleration of database operations and queries for a versioned database based on multiple hardware accelerators" desvela un acelerador de hardware que asiste a un sistema de base de datos huésped que está procesando sus consultas. El acelerador de hardware comprende elementos de procesamiento de objetivo especial que son capaces de recibir tareas de consulta/operación de base de datos en la forma de instrucciones de base de datos de código de máquina,

ejecutar, entonces, en un hardware sin software, y devolver el resultado de la consulta/operación de vuelta al sistema huésped. Por lo tanto, el sistema se refiere a sistemas de base de datos que se optimizan usando aceleración pura de hardware. Una solicitud de patente de Estados Unidos US2003229640A1 desvela un aparato, producto de programa y un procedimiento que utiliza un búfer de consulta dinámicamente rellenada para facilitar el maneo de al menos una parte de la consulta de base de datos en paralelo. Una consulta se implementa usando, al menos, una primera y una segunda parte, donde la segunda parte de la consulta se ejecuta en paralelo usando una pluralidad de procesos. La primera parte de la consulta se ejecuta para rellenar dinámicamente un búfer de consulta con registros desde una fuente de datos, y la pluralidad de procesos que ejecutan la segunda parte de la consulta se especifican en el búfer de consulta para que la fuente de datos efectiva para la segunda parte de la consulta comprenda los registros que se rellenan dinámicamente en el búfer de consulta. Teniendo en cuenta el estado de la técnica anterior, existe una necesidad de diseñar e implementar una optimización de ejecución de consulta de base de datos eficiente que sería más eficiente que las consultas de tipo JOIN. En particular, tal optimización debería dirigirse a aumentar el rendimiento de recuperación de datos en bases de datos normalizadas en el nivel 4NF o 5NF y, preferentemente no requerirá componentes de hardware especializados.

El objetivo de la presente invención es un es un procedimiento implementado por ordenador para ejecutar una consulta de base de datos como se define en las reivindicaciones adjuntas.

Éstos y otros objetivos de la invención presentados en el presente documento se logran proporcionando un procedimiento implementado por ordenador para optimización de ejecución de consulta. Más detalles y característica de la presente invención, y naturaleza y diversas ventajas devendrán más evidentes a partir de la descripción detallada siguiente de las realizaciones preferentes mostradas en un dibujo, en los que:

la figura 1 presenta un ejemplo de una estructura normalizada en el nivel 2;
la figura 2 muestra una normalización 3NF para la base de datos de la figura 1;
la figura 3 muestra una normalización 5NF para la base de datos de la figura 1;
la figura 4 presenta un procedimiento para ejecutar una consulta de base de datos de acuerdo con la invención;
la figura 5 muestra una implementación ejemplar del procedimiento mostrado en la figura 4;
las figuras 6 y 7 muestran un sistema de base de datos de tipo diagrama de ideas; y,
la figura 8 muestra un sistema de base de datos ejemplar, para el que el procedimiento presente se ha diseñado por el que puede implementarse.

En la siguiente descripción detallada, los operadores UNION, EXCEPT e INTERSECT se llamarán operadores de conjunto.

La figura 1 presenta un ejemplo de una estructura normalizada en el nivel 2. En el ejemplo de un sistema de base de datos relacional en esta figura, se asume que se recopila información acerca de los empleados. La tabla "Empleados" principal comprenderá información sobre datos personales. Un empleado tiene un identificador 101, un nombre 102, un apellido 103 y vive en una ciudad 104 en una cierta dirección 105.

También se recopila información acerca de las capacidades y aficiones de los empleados de la siguiente manera. Existe una tabla "Aficiones" que comprende la ID 106 Empleado como una clave remota, la descripción de la afición 107 y su nombre 108. De manera similar, existe una tabla "Capacidades" que comprende la ID 109 Empleado como una clave remota, descripción de la capacidad 110 y su nombre 111.

Una normalización 3NF para la base de datos de la figura 1 se ha mostrado en la figura 2. Como se puede ver, la tabla Empleados sigue siendo la misma, pero las tablas aficiones y capacidades se han combinado en la tabla Hobbies_Skills, que comprende columnas de tanto la tabla de Aficiones como de Capacidades. Por lo tanto, hay un número de tablas que se ha reducido por uno, pero la adición de nuevas capacidades o aficiones para un empleado particular requerirá duplicar un registro en la tabla Hobbies_Skills.

Una normalización 5NF, como se muestra en la figura 3, traería los siguientes cambios. La tabla Empleado sigue siendo la misma, mientras que, en comparación con la tabla 3NF, la tabla Hobbies_Skills se ha dividido en cuatro tablas diferentes. Se han creado dos tablas de puramente de indexación de Employee_Hobby y Employee_Skill, que solo comprende una clave Employee_ID y un índice 112, 113 local de la afición o capacidad, respectivamente. Esas dos tablas son enlaces entre las tablas Aficiones y Capacidades, que, en comparación a las tablas originales de la figura 1 no comprenden ningún dato de la tabla Empleados.

Como ya se ha explicado, conforme el nivel de normalización aumenta, el esfuerzo computacional requerido con el fin de acceder y filtrar los datos también aumenta drásticamente. En primer lugar, el tiempo requerido en el servidor de base de datos para obtener los datos aumenta. Segundo, los datos se designan de tal manera que cuando los resultados ya están localmente listos, la base de datos informa al cliente de cómo acceder al primer elemento del conjunto de resultados y entonces el cliente puede recuperar de manera interactiva todos los registros del conjunto de resultados. Típicamente, esto se ejecuta por un registro, con el fin de ahorrar ancho de banda del medio de comunicación entre el cliente y el servidor. Este enfoque surge del hecho de que incluso aunque un conjunto de resultados puede ser grande, no todos los datos en él necesitan estar preparados para la presentación.

Por lo tanto, cualquier mejora relacionada con minimizar el esfuerzo informático en el servidor y/o reducir el tiempo

de recuperación entre el cliente y el servidor de la base de datos son cruciales.

La figura 4 representa un procedimiento para ejecutar una consulta de base de datos de acuerdo con la invención. Asumamos que en un simple ejemplo de la figura 1 uno necesita extraer os empleados en una ciudad determinada. Este simple ejemplo dará como resultado en la optimización del tiempo de entrega de los lados más que en el tiempo de procesamiento de la consulta en el servidor.

En el resto de esta memoria descriptiva, una columna de una tabla es equivalente a una propiedad de un objeto de una estructura de datos dada (un conjunto).

En la etapa 401, el sistema necesita establecer, qué columna/propiedad de una estructura/tabla de datos comprende el valor único más pequeño para registros dados. No es el valor real de datos, sino más bien el tipo de datos en tal columna/propiedad lo que se toma en cuenta. El tamaño de valores se determina típicamente como tamaño de campo de datos. Por ejemplo, los valores tales como números enteros de 64 bits son mayores que los valores tales como los números enteros de 16 bits.

En el ejemplo de la figura 1, la columna de Identificación comprende los valores únicos más pequeños (es decir, un número entero que es típicamente 32/64 bits frente a una Cadena que es típicamente de 256 bytes). En la vida real existen muchos casos en los que un registro comprende varios valores únicos dentro de un conjunto/tabla/estructura. Por ejemplo, un artículo comercial comprende un identificador, un código QR (del inglés *Quick Response*) y un número de fabricación que son todos únicos para un artículo dado.

La cuestión de si un valor de tipo de datos de columna único dado es más pequeño que otro valor único puede responderse usando un esquema de base de datos.

El esquema de base de datos es una estructura lógica de una base de datos que se define típicamente en un DBMS (Sistema de Administración de Bases de Datos, del inglés *Database Management System*) con un uso de un lenguaje especial, por ejemplo, un DDL (un Lenguaje de Descripción de Datos, del inglés *Data Description Language*). La estructura de base de datos interna se define como metadatos (este término se refiere a "datos sobre datos") en un esquema llamado de información. En bases de datos relacionales, el esquema de información son datos que proporcionan información acerca de todas las tablas y/o vistas y/o columnas y/o procedimientos en una base de datos. Este conjunto de datos en dependiente de la base de datos y sus procedimientos de acceso normalmente son dependientes de la base de datos. Es, sin embargo, común para una base de datos proporcionar tal estructura y acceso a ella. Los metadatos proporcionarán información acerca de los tipos de datos en cada columna/propiedad de cada tabla. Estos tipos pueden compararse con el fin de encontrar la columna/propiedad de valor único más pequeño.

La identificación, en la etapa 401, del valor único más pequeño que se puede recuperar para un registro permite cantidad educida de datos necesarios que transferirán desde un servidor de base de datos hasta un cliente de base de datos.

En la etapa 402, se ejecuta una consulta, que incluye cualquier parámetro de limitación (un filtro de datos), que recuperará solo los datos desde la columna/propiedad que comprende el valor único más pequeño que se puede recuperar para un registro. Por tal enfoque, uno reduce al mínimo la cantidad de datos que se recuperarán.

A continuación, en la etapa 403, comienza la recuperación de resultados. Tan pronto como un número predefinido de resultados (o todos los resultados disponibles) se ha recuperado, el sistema comprueba si el fin del conjunto de resultados se ha alcanzado 404.

Típicamente, el número predefinido de resultados permanecerá en un intervalo de entre 75 a 150 ya que hay una limitación de base de datos de una caché de consulta y las consultas que exceden consulta elementos 150 puede devenir demasiado grande para ajustarse a un único búfer de consulta, aumentando así el esfuerzo computacional requerido por el procesamiento de la consulta. Por otra parte, dividir el resultado en conjuntos de varios elementos aumentaría significativamente la sobrecarga en los procesos de gestión.

Una tabla hash se crea preferentemente donde los valores únicos más pequeños de la consulta de la sección 402 son claves y los resultados de las consultas de las secciones 405 y 406 son valores de la tabla hash.

Si hay más registros en el conjunto de resultados, en la etapa 405 use crea un nuevo proceso de acceso a la base de datos que recuperará los datos que están presentes en los registros identificados con el valor único más pequeño seleccionado en la etapa 401. De lo contrario, si no hay más resultados en el conjunto de resultados, el proceso recupera los registros restantes en la etapa 406 donde un proceso final se crea que recuperará datos de los registros restantes identificados.

Una vez que el proceso 405 comienza, el procedimiento avanza a la etapa 403, donde otro lote de resultados de la sección 402 de consulta se recupera, mientras que el proceso de la sección 405 opera.

Cuando un conjunto de resultados se obtiene para la consulta ejecutada en un proceso 405, 406 de recuperación de

datos, los datos se leen en la tabla hash definida previamente.

Es, por lo tanto, evidente que puede haber una pluralidad de procesos iniciados en la etapa 405, que pueden ejecutarse en paralelo.

Por lo tanto, los resultados finales se obtienen usando procesamiento paralelo y la etapa inicial de recuperación de datos que son los datos representativos más pequeños del conjunto de resultados completo.

Con el fin de mejorar adicionalmente la eficacia en cada proceso 405, 406, un operador establecido, en este caso el operador UNION se usa en una consulta, que combina el resultado de dos o más declaraciones SELECT. La declaración UNION tiene varios requisitos, que son que cada declaración SELECT dentro de la UNION debe tener el mismo número de columnas, y las columnas deben tener tipos de datos similares a las columnas en cada declaración SELECT debe estar en el mismo orden.

El uso de un operador UNION incita al motor del servidor de base de datos al diferente enfoque de procesamiento de consultas que en el caso de un operador OR típico entre diferentes valores o parámetros. En el caso de UNION hay consultas internas separadas de parámetros separados de una única consulta.

Por ejemplo, una consulta ejecutada en un proceso puede tener una forma similar a uno definido basándose en la tabla Empleados de la figura 1:

```
SELECT Employees.* FROM Employees WHERE ID=3 UNION
SELECT Employees.* FROM Employees WHERE ID=6
UNION
SELECT Employees.* FROM Employees WHERE ID=16 UNION
```

El proceso de ejecutar una consulta 402, que incluye cualquier parámetro de limitación, puede optimizarse adicionalmente, especialmente en caso en el que la selección de valores particulares de la columna/propiedad que comprende el valor único más pequeño requiere consultar numerosas tablas/estructuras de datos.

En 4NF o 5NF, las consultas típicas de tipo JOIN a menudo son tan complejas que sus ejecuciones toman horas de procesamiento de datos extensivo.

Por ejemplo, uno necesita seleccionar empleados a los que les gusta el ciclismo (tabla de aficiones separada) y que se promocionaron durante los dos últimos años (tabla de promociones). En tales casos, en lugar de usar consultas de tipo JOIN uno puede emplear un operador INTERSECT. El operador INTERSECT se usa entre dos consultas SELECT seleccionadas con el fin de devolver solo valores que coinciden dentro de ambos conjuntos de datos devueltos por las consultas SELECT respectivas.

Por ejemplo:

```
SELECT Employee.ID FROM Employees WHERE Employee.NAME =
'ADAM'
INTERSECT
SELECT Hobbies.Employee_ID FROM Hobbies WHERE Hobbies.Name =
'cycling';
```

Esto devolverá una lista de identificadores de empleados cuyo nombre es Adam y a los que, al mismo tiempo, les gusta el ciclismo. Tal enfoque de utilización de INTERSECT y no JOIN con el fin de encontrar los valores únicos más pequeños es muy eficiente.

El uso de consultas de intersección es mucho más eficiente para consultar estructura cruzada y la presente invención devuelve los mejores resultados, en términos de tiempo de recuperación de datos, en caso de consultas de estructura cruzada compleja.

La figura 5 muestra una implementación ejemplar del procedimiento mostrado en la figura 4 por simplicidad y presentación de una idea general que puede implementarse en diferentes lenguajes de programación, el presente ejemplo se formula en pseudocódigo.

El sistema presentado en el presente documento es especialmente útil en bases de datos basándose en diagramas de ideas tal como una base de datos desvelada en la Solicitud de Patente Europea pendiente de aprobación número EP13461516.0 por el mismo Solicitante. En qué tipo de base de datos particular, es especialmente fácil ejecutar un procedimiento para encontrar objetos de interés que se relacionan a diferentes estructuras de datos porque, una relación entre objetos está presente en la estructura de datos OBJECT RELATIONS.

El uso de consultas que operan sobre conjuntos de datos, es decir, el uso de operadores de conjunto, en una base de datos de tipo esquema de ideas proporciona resultados mejorados debido a la estructura de la base de datos. Cada consulta que tendrá en cuenta descripciones de objeto, objetos, relaciones, columnas y conjuntos requeriría cada vez formular una consulta, que debido a la heterogeneidad de los tamaños de las tablas sería altamente

ineficiente, especialmente cuando se ejecutan con frecuencia.

Los objetos y sus descripciones en estructuras de una base de datos de tipo esquema de ideas, comprende un identificador de objeto. Por lo tanto, cuando se busca un objeto uno puede limitar no solo la estructura de datos característicos de objeto, es decir, la cuarta estructura 701 de datos, ya que esta estructura identifica sin ambigüedad un objeto. Esto tiene una especial ventaja durante la búsqueda de objetos que se relacionen con otros objetos de un conjunto diferente. Tal búsqueda puede formularse como sigue:

- para objetos seleccionados y una relación seleccionada, ejecutar una función de búsqueda sobre relaciones 708 de objeto con un identificador de relación seleccionado
- desde la estructura de características, seleccionar, por medio de otra función de búsqueda, solo aquellos que cumplen la condición, por ejemplo, en la columna 13 hay un valor de "ADAM";
- Combinar ambas consultas con un operador INTERSECT.

Añadir otra restricción relacionada con un valor diferente en una columna diferente es meramente una adición de una consulta simple y que limita los resultados con un uso de operador INTERSECT.

Lo más importante es que el proceso de búsqueda y de navegación completos de los resultados se centra en varias consultas que permanecen inalteradas excepto por los valores específicos de parámetros, debido a lo que no hay necesidad de escribir código de programación redundante con el fin de cubrir más consultas y más específicas.

Debido al uso de UNION e INTERSECT, existe una posibilidad de procesamiento mejorado y más rápido de consultas.

Otro aspecto relacionado con UNION, INTERSECT y EXCEPT es que las estructuras de datos fundamentalmente diferentes pueden consultarse para los mismos datos.

La siguiente sección de la memoria descriptiva presenta una parte clave de la Solicitud de Patente Europea pendiente de aprobación número EP13461516.0 del Solicitante.

La figura 6, que corresponde a la figura 2 de la solicitud pendiente de aprobación, muestra un nuevo sistema de base de datos de acuerdo con la presente invención. Con el fin de cooperar perfectamente con esquemas de ideas, el sistema de base de datos de acuerdo con la invención se ha diseñado de diferente manera a los sistemas de bases de datos conocidos. El sistema de base de datos comprende seis conjuntos de núcleo de datos y conjuntos opcionales. Los conjuntos de núcleos comprenden CONJUNTOS, OBJETOS, COLUMNAS, CARACTERÍSTICAS, RELACIONES y RELACIONES DE OBJETOS. Cabe señalar que los nombres anteriores son ejemplares solo y que los conjuntos de núcleos respectivos se definen más por su función dentro del sistema que por su nombre.

El primer conjunto de datos se llama CONJUNTOS 604, porque se usa para mantener lógicamente datos relacionados con los conjuntos de datos. Los conjuntos de datos pueden representarse sobre un esquema de ideas como nodos. Cada entrada en la estructura 604 de datos CONJUNTOS comprende, al menos, un único identificador 605a y preferentemente también su nombre 605. En referencia, de nuevo, al ejemplo de la figura 1, existen tres CONJUNTOS, es decir, COLORES que tienen identificación de 1, MATERIALES que tienen identificación de 2 y HERRAMIENTAS que tienen identificación de 3. La estructura de datos CONJUNTOS es una estructura de nivel superior y no se refiere a otras estructuras de datos, sino que otras estructuras de datos se refieren a ello como identificándose por las flechas respectivas entre los conjuntos de la figura 6.

Cada conjunto de datos, como en el mundo real, se caracteriza por algunas propiedades típicamente llamadas columnas. Por lo tanto, el segundo conjunto de datos se llama COLUMNAS 606. Una propiedad, llamada típicamente una columna, se identifica únicamente con un identificador ID 607 y se asocia con un conjunto, definido en la estructura 604 de datos CONJUNTOS, por medio de un identificador llamado en el presente documento ID DE CONJUNTO 608. Una columna también se asocia preferentemente con un nombre 609. Como se indica por una flecha 604a, la estructura de datos COLUMNAS lógicamente, se refiere directamente a la estructura de datos CONJUNTOS, porque usa los identificadores de conjuntos. Si, por ejemplo, cada color del conjunto llamado COLORES tiene otra propiedad, digamos, valor RGB, podría añadirse a una entrada que comprende los siguientes valores: "1", "4", "RGB". En este nivel del sistema, los tipos de datos respectivos tales como texto, número entero, BLOB no se consideran como su aplicación en el presente sistema es trabajo de rutina.

Habiendo definido las estructuras de datos de CONJUNTOS y COLUMNAS se pueden definir objetos que formarán elementos de CONJUNTOS respectivos y tendrán propiedades definidas por las estructuras de datos COLUMNAS. Los objetos se mantienen en la estructura de datos OBJETOS 601. Esta estructura de datos mantiene únicamente entradas identificadas con un identificador ID 603 y un se asocian con un conjunto, definido en la estructura 604 de datos CONJUNTOS, por medio de un identificador llamado en el presente documento ID DE CONJUNTO 602. Como se indica por una flecha 601a, la estructura de datos OBJETOS lógicamente, se refiere directamente a la estructura de datos CONJUNTOS, porque usa los identificadores de conjuntos.

La cuarta estructura de datos núcleo es una estructura de datos que mantiene entradas de datos de cada propiedad de cada objeto. Esta estructura de datos se ha llamado CARACTERÍSTICAS 301 en la figura 6. Esto es una de las

diferencias fundamentales de todas las bases de datos conocidas donde hay filas de datos que comprenden entradas para todas las columnas de una tabla de datos. En la presente invención, cada propiedad de un objeto se almacena como una entrada separada, que mejora enormemente la escalabilidad del sistema y permite, por ejemplo, añadir propiedades de objetos en tiempo real.

La estructura de datos CARACTERÍSTICAS 701 mantiene únicamente entradas identificadas con un identificador ID DE OBJETO 702 y se asocia con una propiedad, definida en la estructura 606 de datos COLUMNAS, por medio de un identificador llamado en el presente documento ID DE COLUMNA 703. Además, cada entrada en la estructura de datos CARACTERÍSTICAS, comprende un valor de la propiedad dada del objeto particular. Como se indica por las flechas respectivas que se originan a partir de las fuentes A y B, la estructura 701 de datos CARACTERÍSTICAS lógicamente, se refiere directamente a estructura de datos COLUMNAS y estructura de datos OBJETOS, porque usa los identificadores desde las estructuras de datos respectivas.

La quinta estructura de datos núcleo, del sistema de base de datos de acuerdo con la presente invención, se diseña para mantener dato respecto a las relaciones presentes en la base de datos. Esta estructura se ha llamado aquí RELATIONS 705. Es una estructura muy simple y, en principio, mantiene un identificador de una relación ID 707 y, preferentemente también mantiene la descripción textual de la relación, es decir, un NAME 706. Como se indica por una flecha 705a, la estructura de datos RELACIONES lógicamente, hace directamente referencia hacia abajo de la estructura de datos RELACIONES DE OBJETOS, porque las RELACIONES DE OBJETOS usan los identificadores de las relaciones.

La última estructura de datos núcleo de la presente invención es la mencionada estructura 708 de datos RELACIONES DE OBJETOS. Esta estructura de datos de diseña para proporcionar un esquema entre una relación de la estructura 705 de datos RELACIONES y dos objetos de la estructura 701 de datos OBJETOS. Por ejemplo, la primera entrada en la estructura 708 de datos define que la relación que tiene identificador de 1existe entre el un objeto que tiene un identificador de 1 y el objeto que tiene un identificador de 6.

Opcionalmente, existe una séptima estructura de datos en el sistema de base de datos de la presente invención. Esta estructura de datos mantiene datos con respecto a las relaciones entre los conjuntos de datos respectivos y en la figura 7 se llama RELACIONES DE CONJUNTOS 712. Esta estructura de datos se diseña para proporcionar un esquema entre una relación desde la estructura 705 de datos RELACIONES y dos conjuntos de estructura 604 de datos CONJUNTOS. Por ejemplo, la primera entrada en la estructura 712 de datos RELACIONES DE CONJUNTOS define que la relación que tiene un identificador de 1 existe entre un conjunto que tiene un identificador de 1 y un conjunto que tiene un identificador de 2. Proporcionar una entrada en la estructura 712 de datos RELACIONES DE CONJUNTOS entre un conjunto que tiene un identificador de 1 y un conjunto que tiene un identificador 2, así como entre un conjunto que tiene un identificador de 2 y un conjunto que tiene un identificador de 1, permite crear una relación bidireccional.

También hay una posibilidad de referencia propia desde un conjunto dado. Por ejemplo, tal caso puede estar presente cuando hay un conjunto de personas y existe una relación de estudiante - profesor entre personas asignadas a un conjunto particular.

Como se describe, por ejemplo, un sistema de base de datos relacional de cien tablas se almacenará en el presente sistema en las seis estructuras de datos descritas anteriormente. Naturalmente, la mayoría de los datos permanecerán en las estructuras de datos OBJETOS y CARACTERÍSTICAS.

Como se puede ver en la base de datos de tipo esquema de ideas, los objetos se relacionan directamente por medio de relaciones de objeto.

La figura 8 muestra un sistema de base de datos ejemplar, para el que el procedimiento presente se ha diseñado por el que puede implementarse. El sistema de base de datos comprende un cliente 801 y un servidor 802. El cliente 801 accede a los datos y, por lo general, es un terminal remoto del servidor 802 que aloja una base 806 de datos y un sistema 807 de gestión de bases de datos responsable de responder las consultas del cliente. El cliente es normalmente un ordenador que comprende una memoria 803, un procesador 804 y un módulo 805 para ejecutar el procedimiento definido en la figura 4. Será evidente que un canal de comunicación adecuado debe establecerse entre el cliente 801 y el servidor 802.

Puede reconocerse fácilmente, por un experto en la materia, que el procedimiento implementado por ordenador anterior para ejecutar consultas de bases de datos puede realizarse y/o controlarse por uno o más programas informáticos. Tales programas informáticos se ejecutan típicamente utilizando recursos informáticos en un dispositivo informático tal como ordenadores personales, asistentes digitales personales, teléfonos móviles, receptores y decodificadores de televisión digital o similares. Las solicitudes se almacenan en memoria no volátil, por ejemplo, una memoria flash o memoria volátil, por ejemplo, RAM, y se ejecutan por un procesador. Estas memorias son medios de grabación ejemplares para almacenar programas informáticos que comprenden instrucciones ejecutables por ordenador que realizan todas las etapas del procedimiento implementado por ordenador de acuerdo con el concepto técnico presentado en el presente documento.

Mientras la invención presentada en el presente documento se ha representado, descrito, y se ha definido con

referencia a realizaciones preferentes particulares, tales referencias y ejemplos de implementación en la memoria descriptiva anterior no implican ninguna limitación de la invención. Será, sin embargo, evidente que diversas modificaciones y cambios pueden realizarse en ello sin alejarse del ámbito más amplio del concepto técnico. Las realizaciones preferentes presentadas son solo ejemplares, y no son exhaustivas del ámbito del concepto técnico presentado en el presente documento.

Por consiguiente, el ámbito de protección no se limita a las realizaciones preferentes descritas en la memoria descriptiva, sino que solo se limita por las reivindicaciones que siguen.

REIVINDICACIONES

1. Un procedimiento implementado por ordenador para ejecutar una consulta de base de datos en una base de datos, comprendiendo el procedimiento las etapas de:

• proporcionar, al menos, una estructura de datos, definida por un esquema de base de datos de dicha base de datos, que comprende, al menos, un objeto que tiene, al menos, dos propiedades, teniendo cada propiedad un tipo de datos asignado, en el que dichos tipos de datos pueden compararse con respecto a su tamaño de datos;
• establecer (401), usando dicho esquema de base de datos de dicha base de datos, por comparación, de dichos tipos de datos de las al menos dos propiedades, una propiedad de una estructura de datos seleccionada que comprende los valores únicos más pequeños para registros dados de tipo de datos almacenados en esa propiedad particular;
• ejecutar (402) la consulta de base de datos, que incluye cualquier parámetro de limitación, configurado para recibir solo datos desde la propiedad que comprende los valores únicos más pequeños que son recuperables para un registro;
• empezar (403) a recuperar resultados de la consulta (402) recuperada;
• para cada conjunto de número de resultados recuperados predefinido, ejecutar (405) un nuevo proceso de acceso a la base de datos que se configura para recuperar datos que están presentes en los registros identificados con los valores únicos más pequeños.

2. El procedimiento de acuerdo con la reivindicación 1 en el que la etapa (401) de establecimiento se basa en la información de esquema de la base de datos.

3. El procedimiento de acuerdo con la reivindicación 1 en el que el número predefinido está entre 75 y 150 y los procesos de acceso a la base de datos se ejecutan en paralelo.

4. El procedimiento de acuerdo con la reivindicación 1 en el que los resultados recuperados se almacenan en una tabla hash en la que los valores únicos más pequeños de la consulta (402) son claves y los resultados de las consultas (405), (406) posteriores son valores de la tabla hash.

5. El procedimiento de acuerdo con la reivindicación 1 en el que la consulta ejecutada por cada proceso (405), (406) utiliza los operadores UNION entre las consultas SELECT limitadas con los valores de la propiedad de valor único más pequeña.

6. El procedimiento de acuerdo con la reivindicación 1 en el que la etapa de ejecutar una consulta (402) utiliza, en caso de estructuras de datos de numerosas consultas cruzadas, un operador INTERSECT entre subconsultas relacionadas con diferentes estructuras de datos.

7. El procedimiento de acuerdo con la reivindicación 1 en el que la base de datos comprende:

• una primera estructura (604) de datos, almacenada en la memoria, que comprende una definición de al menos un conjunto de datos en la que cada conjunto de datos comprende un identificador de conjunto de datos y contiene de manera lógica objetos de datos del mismo tipo;
• una segunda estructura (606) de datos, almacenada en la memoria, que comprende definiciones de propiedades de objetos en las que cada propiedad comprende un identificador de la propiedad y un identificador de un conjunto, desde la primera estructura (604) de datos, a la que la propiedad se asigna;
• una tercera estructura (601) de datos, almacenada en la memoria, que comprende definiciones de objetos en la que cada objeto comprende un identificador y un identificador de un conjunto, desde la primera estructura (604) de datos, a la que el objeto se asigna;
• una cuarta estructura (701) de datos, almacenada en la memoria, que comprende definiciones de propiedades de cada objeto en la que cada propiedad de un objeto asocia un valor con un objeto, desde la tercera estructura (601) de datos, y una propiedad del conjunto,desde la segunda estructura (606) de datos, a la que el objeto se asigna;
• una quinta estructura (705) de datos, almacenada en la memoria, que comprende definiciones de relaciones en la que cada relación comprende un identificador de relación; y
• una sexta estructura (708) de datos, almacenada en la memoria, para almacenar definiciones de relaciones entre objetos en la que cada relación de objetos se asocia una relación, desde la quinta estructura (708) de datos, hasta dos objetos desde la tercera estructura (601) de datos.

8. Un programa informático que comprende medios de código de programa para realizar todas las etapas del procedimiento implementado por ordenador de acuerdo con cualquiera de las reivindicaciones 1 - 7 cuando dicho programa se ejecuta en un ordenador.

9. Un medio legible por ordenador que almacena instrucciones ejecutables por ordenador que realizan todas las etapas del procedimiento implementado por ordenador de acuerdo con cualquiera de las reivindicaciones 1 - 7 cuando se ejecuta en un ordenador.

101 ID	102 NOMBRE	103 APELLIDO	104 CIUDAD	105 Dirección
1	A	A	C1	A1
2	B	B	C2	A2
3	C	C	C3	A3

Empleados

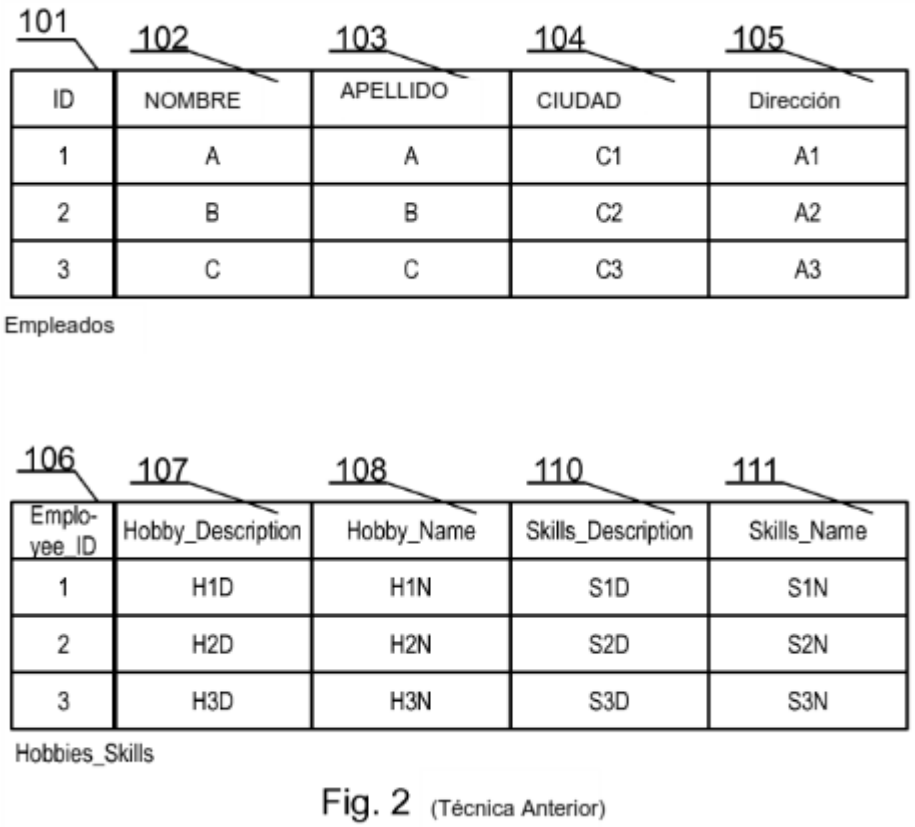
106 Emplo-yee_ID	107 Descripción	108 Nombre
1	H1D	H1N
2	H2D	H2N
3	H3D	H3N

Aficiones

109 Emplo-yee_ID	110 Descripción	111 Nombre
1	S1D	S1N
2	S2D	S2N
3	S3D	S3N

Capacidades

Fig. 1 (Técnica Anterior)



101

ID	NOMBRE	APELLIDO	CIUDAD	Dirección
1	A	A	C1	A1
2	B	B	C2	A2
3	C	C	C3	A3

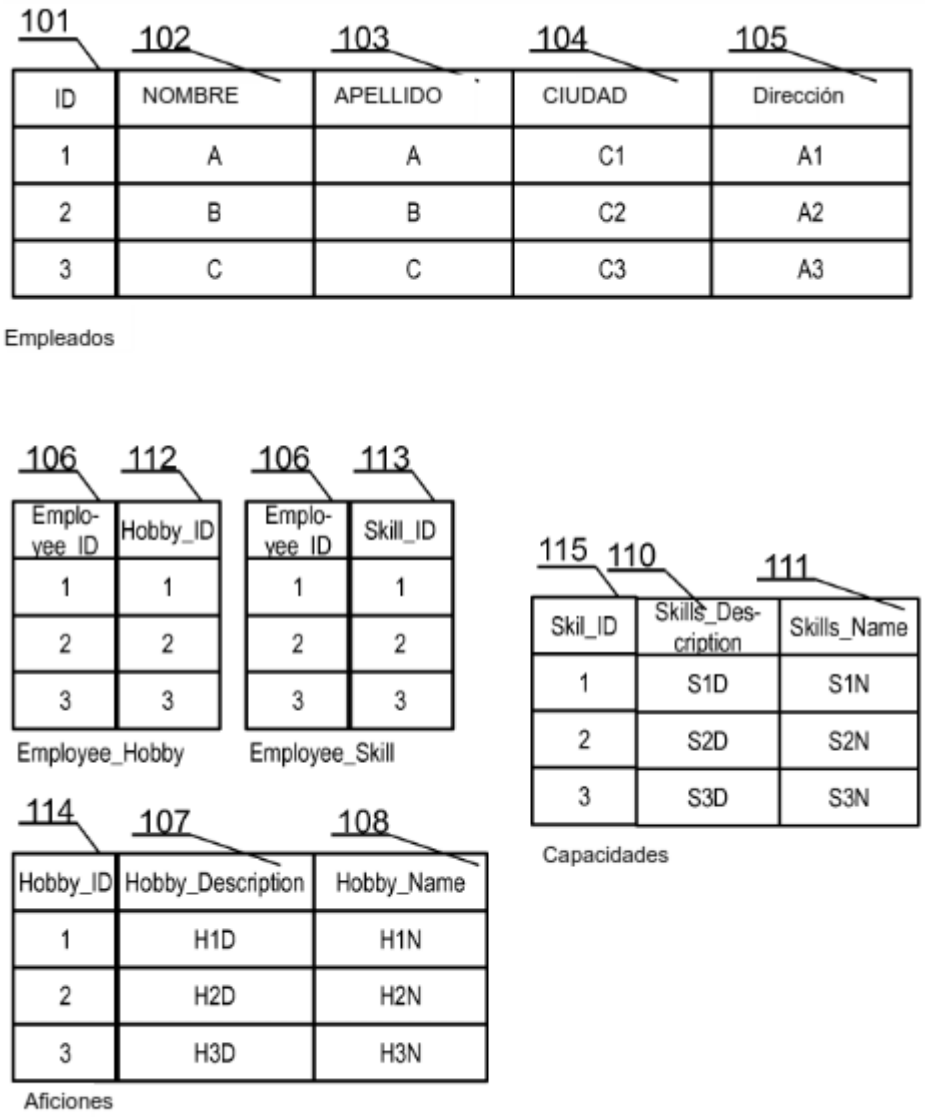
Empleados

106

Emplo-yee_ID	Hobby_Description	Hobby_Name	Skills_Description	Skills_Name
1	H1D	H1N	S1D	S1N
2	H2D	H2N	S2D	S2N
3	H3D	H3N	S3D	S3N

Hobbies_Skills

Fig. 2 (Técnica Anterior)



101 102 103 104 105

ID	NOMBRE	APELLIDO	CIUDAD	Dirección
1	A	A	C1	A1
2	B	B	C2	A2
3	C	C	C3	A3

Empleados

106 112

Emplo-yee_ID	Hobby_ID
1	1
2	2
3	3

Employee_Hobby

106 113

Emplo-yee_ID	Skill_ID
1	1
2	2
3	3

Employee_Skill

115 110 111

Skil_ID	Skills_Des-cription	Skills_Name
1	S1D	S1N
2	S2D	S2N
3	S3D	S3N

Capacidades

114 107 108

Hobby_ID	Hobby_Description	Hobby_Name
1	H1D	H1N
2	H2D	H2N
3	H3D	H3N

Aficiones

Fig. 3 (Técnica Anterior)

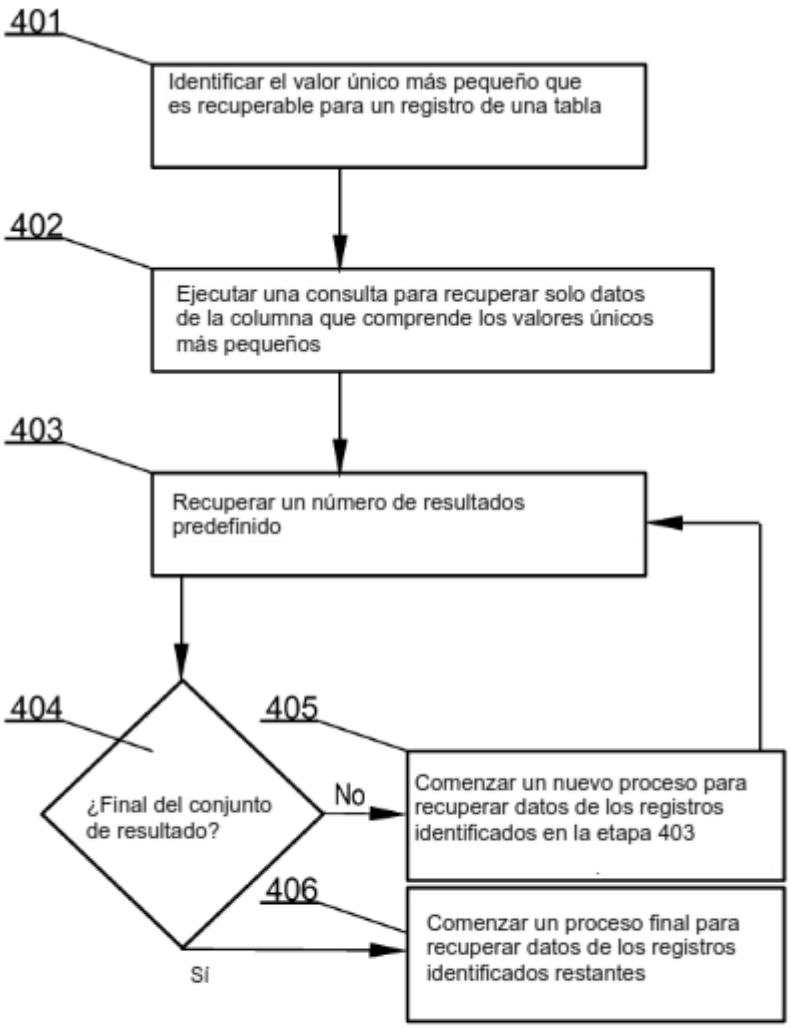

401
Identificar el valor único más pequeño que es recuperable para un registro de una tabla
402
Ejecutar una consulta para recuperar solo datos de la columna que comprende los valores únicos más pequeños
403
Recuperar un número de resultados predefinido
404
¿Final del conjunto de resultado?
No
Sí
405
Comenzar un nuevo proceso para recuperar datos de los registros identificados en la etapa 403
406
Comenzar un proceso final para recuperar datos de los registros identificados restantes

Fig. 4

```
String tableName = "TableName";
String smallestColumnName = getSmallestUniqueValueForTable(tableName);
String query = "";
if ( defined filters exists )
        while ( filters has next )
                if ( query is not empty ) {
                        query += filters.get().getSetOperator();
                query += filter.getSubQuery();
smallestUniqValueResultSet = executeQuery(query);
integer numberOfResults = 100;
integer executeThreadsNumber = 5;
while( smallestUniqValueResultSet has more results )
        Results results = smallestUniqValueResultSet.getPart(numberOfResults);
        fetchWholeRowInOtherThread(results);
```

Fig. 5

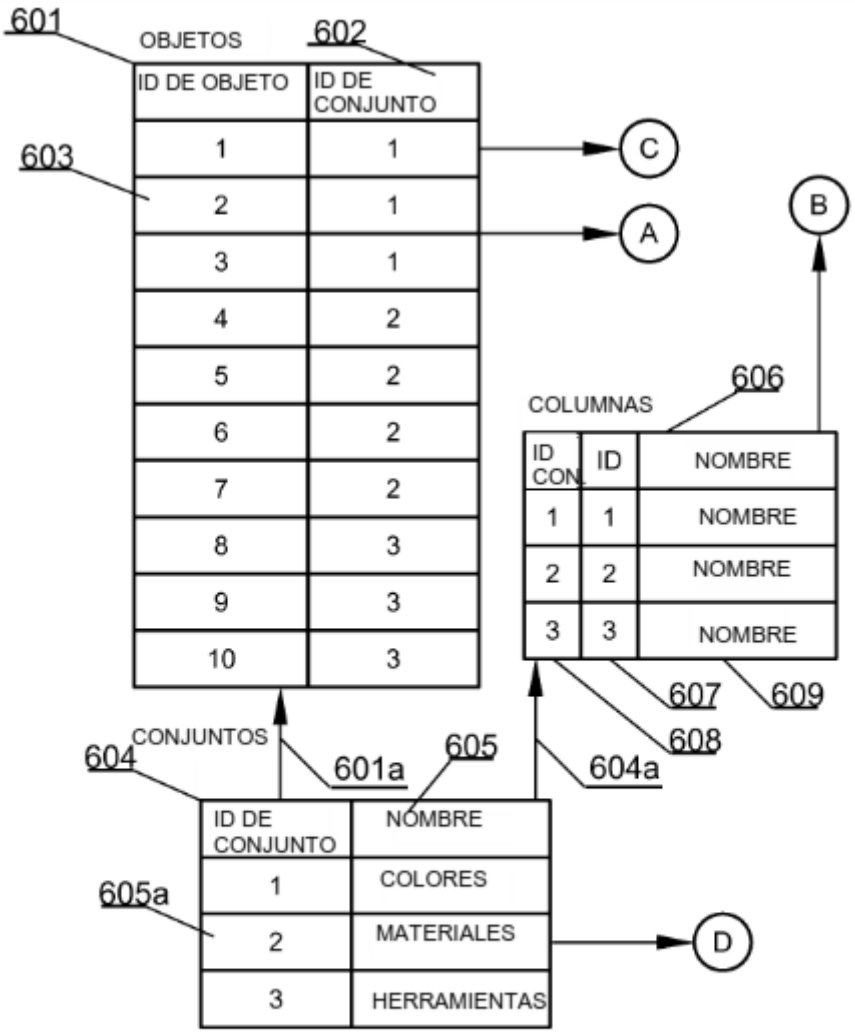

601
OBJETOS
602
ID DE OBJETO
ID DE CONJUNTO
603
1 1
2 1
3 1
4 2
5 2
6 2
7 2
8 3
9 3
10 3
C
A
B
606
COLUMNAS
ID CON.
ID
NOMBRE
1 1 NOMBRE
2 2 NOMBRE
3 3 NOMBRE
607
609
608
CONJUNTOS
604
601a
605
604a
ID DE CONJUNTO
NOMBRE
605a
1 COLORES
2 MATERIALES
3 HERRAMIENTAS
D

Fig. 6

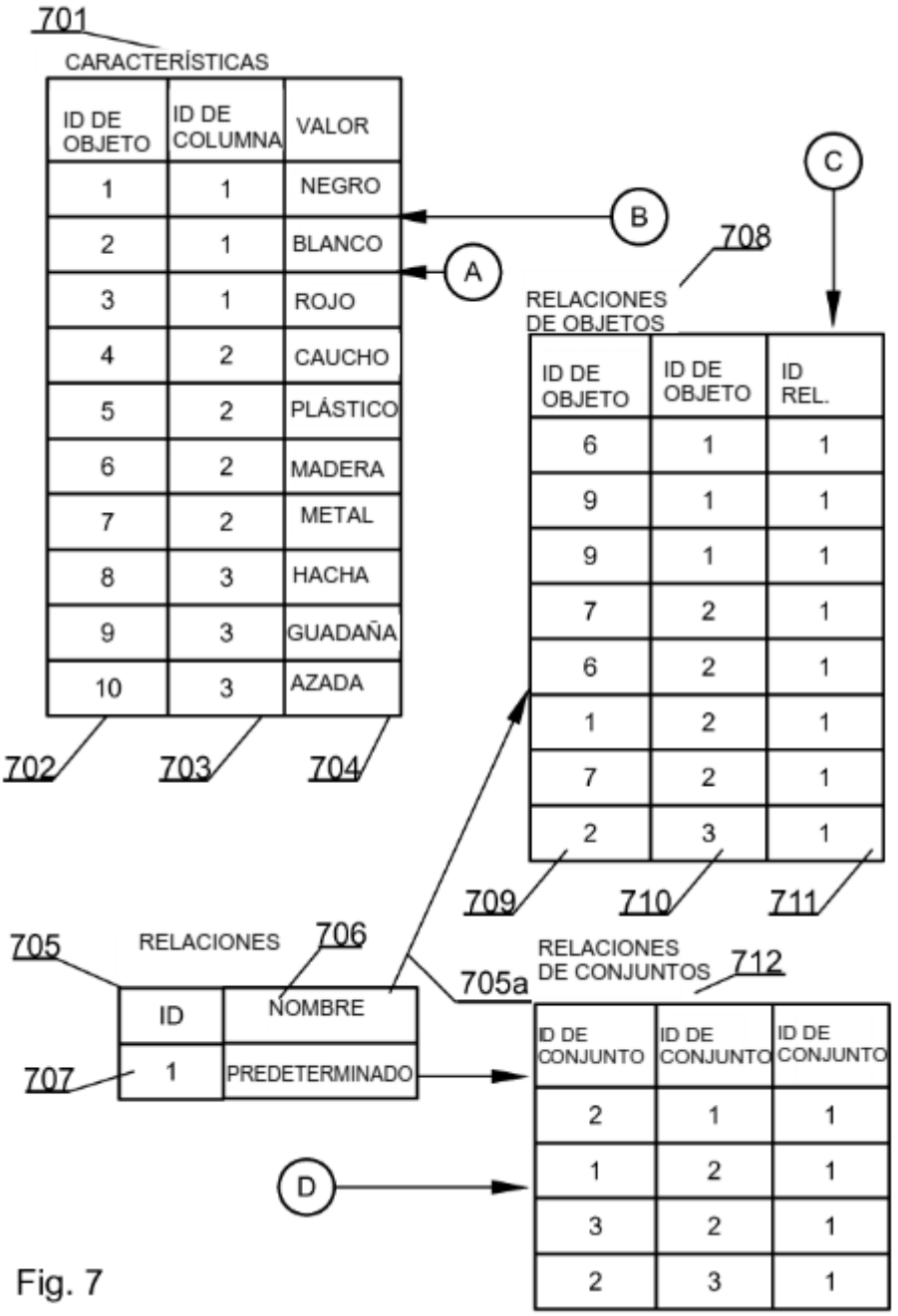

701
CARACTERÍSTICAS
ID DE OBJETO
ID DE COLUMNA
VALOR
1 1 NEGRO
2 1 BLANCO
3 1 ROJO
4 2 CAUCHO
5 2 PLÁSTICO
6 2 MADERA
7 2 METAL
8 3 HACHA
9 3 GUADAÑA
10 3 AZADA
702
703
704
A
B
C
708
RELACIONES DE OBJETOS
ID DE OBJETO
ID DE OBJETO
ID REL.
6 1 1
9 1 1
9 1 1
7 2 1
6 2 1
1 2 1
7 2 1
2 3 1
709
710
711
705
RELACIONES
706
ID
NOMBRE
707
1 PREDETERMINADO
705a
RELACIONES DE CONJUNTOS
712
ID DE CONJUNTO
ID DE CONJUNTO
ID DE CONJUNTO
2 1 1
1 2 1
3 2 1
2 3 1
D

Fig. 7

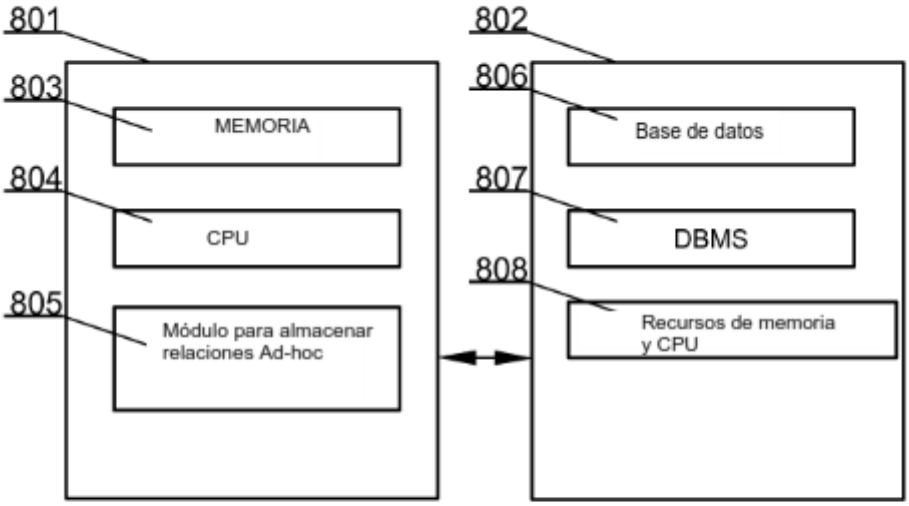

801
802
803
MEMORIA
806
Base de datos
804
CPU
807
DBMS
808
805
Módulo para almacenar relaciones Ad-hoc
Recursos de memoria y CPU

Fig. 8