(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 387 625**

(51) Int. CI.:
H04L 29/12 (2006.01)
H04L 29/08 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Número de solicitud europea: **07150067 .2**
(96) Fecha de presentación: **17.12.2007**
(97) Número de publicación de la solicitud: **2073505**
(97) Fecha de publicación de la solicitud: **24.06.2009**

(54) Título: **Encaminamiento de consulta en un sistema de base de datos distribuida**

(45) Fecha de publicación de la mención BOPI:
27.09.2012

(45) Fecha de la publicación del folleto de la patente:
27.09.2012

(73) Titular/es:
NOKIA SIEMENS NETWORKS OY
KARAPORTTI 3
02610 ESPOO, FI

(72) Inventor/es:
Andersen, Frank-Uwe;
Hoßfeld, Tobias;
Neitzert, Gerald;
Oechsner, Simon;
Scheidl, Wolfgang;
Schoen, Hans-Ulrich;
Tran-Gia, Phuoc y
Tutschku, Kurt

(74) Agente/Representante:
Zuazo Araluze, Alexander

DESCRIPCIÓN

Encaminamiento de consulta en un sistema de base de datos distribuida.

Campo de la invención

La presente invención se refiere a métodos y servidores aplicables en sistemas de servidor de aplicación o base de datos distribuida, en particular en una capa *front-end* de tales sistemas.

Antecedentes de la invención

En un sistema de base de datos distribuida los datos se ubican no sólo en uno sino en varios servidores de base de datos diferentes. Se requiere un sistema de búsqueda con el fin de poder encaminar las consultas para un elemento de datos específico a un servidor de base de datos denominado servidor de base de datos *back-end* que almacena este elemento. Los punteros a estos servidores de base de datos *back-end* se almacenan en el sistema de búsqueda que también se denomina sistema *front-end*. Estos punteros son generalmente del formato <clave, valor>, donde la clave es una clave de búsqueda válida para un servidor de base de datos *back-end*, y el valor es una dirección del servidor de base de datos *back-end* en la que se ubica el correspondiente elemento de datos. Por tanto, cuando se emite una consulta, por ejemplo, una orden de LDAP (protocolo ligero de acceso al directorio), a un sistema de base de datos distribuida, se resuelve en primer lugar en el sistema *front-end* (que está constituido por servidores *front-end*, también denominados a continuación en el presente documento servidores de un primer tipo de servidor), y luego se reenvían a la dirección correcta en el sistema *back-end* (que está constituido por servidores *back-end*, también denominados a continuación en el presente documento servidores de un segundo tipo de servidor), donde puede procesarse.

Recientemente, existe la tendencia en el diseño de sistemas de bases de datos de redes móviles de crear una sola base de datos lógica que físicamente se ejecuta en múltiples servidores y también unifica diferentes tipos de bases de datos, por ejemplo, HLR (registro de ubicación local), RADIUS (servidor de autenticación remota de marcación de usuario), DB de OAM (base de datos de operación y mantenimiento), DB de NM (base de datos de gestión de red), etc.

En los sistemas *front-end* de base de datos distribuida convencionales, los punteros descritos anteriormente se guardan en todos los nodos que constituyen el sistema *front-end*.

El documento US 6.351.775 B1 da a conocer un equilibrio de carga por los servidores en una red informática, en la que, para el encaminamiento dinámico de peticiones de objeto, un método de partición puede mapear los identificadores de objeto con clases y los nodos solicitantes mantienen una tabla de asignación de servidor para mapear cada clase con una selección de servidor.

Sumario de la invención

La invención pretende proporcionar un encaminamiento de consultas de base de datos eficaz en memoria, en un sistema de base de datos distribuida.

Esto se logra mediante métodos y servidores tal como se definen en las reivindicaciones adjuntas. La invención también puede implementarse como un producto de programa informático.

Según la invención, en lugar de almacenar una tabla de búsqueda completa (es decir, una recopilación de punteros, que indiquen dónde se ubican los elementos de datos en dispositivos de servidor *back-end*) en cada dispositivo de servidor *front-end*, cada dispositivo de servidor *front-end* sólo necesita mantener una parte de la tabla de búsqueda (denominada una respectiva segunda tabla de búsqueda) y una “tabla de partición” (denominada una primera tabla de búsqueda) que indica qué dispositivo de servidor *front-end* es responsable de qué sección de datos almacenada en los dispositivos de servidor *back-end*, es decir, qué parte de la tabla de búsqueda se ubica en qué dispositivo de servidor *front-end*.

Según una realización de la invención, se recibe un mensaje de consulta por un dispositivo de servidor *front-end* que se denomina un primer dispositivo. A parte de los datos en el mensaje de consulta se les aplica un *hash* en el primer dispositivo de servidor. Comparando el resultado de la aplicación de un *hash* y la tabla de partición (primera tabla de búsqueda), el primer dispositivo de servidor detecta dónde se ubica un puntero de datos solicitados.

Si el puntero de los datos solicitados se ubica en el primer dispositivo de servidor, envía una petición a un dispositivo de servidor *back-end* indicado por el puntero para extraer los datos.

Si el puntero de los datos solicitados no está ubicado en el primer dispositivo de servidor, según la tabla de partición, el primer dispositivo conoce el dispositivo de servidor *front-end* correcto (segundo dispositivo de servidor) que contiene el puntero de los datos solicitados. El primer dispositivo de servidor en tal caso reenvía el mensaje de

consulta al segundo dispositivo de servidor.

El segundo dispositivo de servidor resuelve el mensaje de consulta sustancialmente de la misma forma que el primer dispositivo (pero sin consultar la primera tabla de búsqueda en el segundo servidor) y devuelve el puntero de los datos solicitados al primer dispositivo.

El primer dispositivo de servidor envía entonces una petición a un servidor *back-end* indicado por el puntero y extrae los datos.

El primer dispositivo de servidor contesta al mensaje de consulta con los datos solicitados.

Según la presente invención, en el peor de los casos, sólo se necesita un reenvío para ubicar los datos solicitados. Cada dispositivo de servidor *front-end* sólo necesita almacenar la tabla de partición y una parte de la tabla de búsqueda en lugar de la totalidad de la tabla de búsqueda, lo que ahorra bastante memoria.

La presente invención es útil especialmente en grandes escenarios de despliegue, en los que existe una gran cantidad de dispositivos de servidor *front-end*.

La presente invención puede aplicarse a una capa *front-end* de servicios o aplicaciones de base de datos a gran escala. Las consultas de aplicación destinadas a una capa *back-end* que contiene conjuntos de datos solicitados o una lógica de servicio se envían a través de la capa *front-end* con el fin de distribuir la carga, procesar previamente los datos o encaminar consultas según la disposición física del sistema *back-end*.

En una aplicación crítica en el tiempo, por ejemplo, el acceso a datos de usuario móvil como un proveedor, la cantidad de tiempo invertido en los dispositivos de servidor de la capa *front-end* debe ser lo más corta posible. Por tanto, los datos de búsqueda (contenidos en la primera y/o respectiva segunda tabla de búsqueda) se almacenan en la memoria de acceso aleatorio (RAM) de un respectivo dispositivo de servidor *front-end*, desde la que puede accederse más rápido en comparación con un escenario en el que se almacenan en un disco duro.

Esto hace que las limitaciones de hardware, es decir, con cuánta memoria puede equiparse un servidor, sean un problema. Con la cantidad de datos almacenada en el *back-end*, el número de entradas de búsqueda en el *front-end* también aumenta. Lo mismo se aplica para un número superior de claves de búsqueda. Por tanto, la cantidad de datos de búsqueda puede volverse demasiado grande para su almacenamiento en un único servidor.

Un servidor *front-end* según la presente invención sólo necesita almacenar una pequeña tabla, es decir, una primera tabla de búsqueda que indique qué servidor *front-end* es responsable de qué sección de datos almacenada en los servidores *back-end*, y una parte de los datos de búsqueda en una respectiva segunda tabla de búsqueda. Por tanto, la invención puede hacer frente a aplicaciones críticas en el tiempo en grandes escenarios de despliegue.

Breve descripción de los dibujos

La figura 1 muestra un diagrama de bloques esquemático que ilustra una disposición de un sistema según una realización de la invención.

La figura 2 que consiste en las figuras 2A y 2B muestra un diagrama de flujo que ilustra un método de consulta de base de datos según una realización de la invención.

La figura 3 muestra un diagrama de flujo que ilustra un método de consulta de base de datos según otra realización de la invención.

La figura 4 muestra el mapeo entre secciones de datos almacenados en dispositivos de servidor *back-end* y dispositivos de servidor *front-end* según una realización de la invención.

La figura 5 muestra un diagrama esquemático que ilustra el encaminamiento de consulta según una realización de la invención.

La figura 6 muestra un diagrama esquemático que ilustra la colocación de los datos y el servidor en un espacio de identificador según una realización de la invención.

La figura 7 muestra un diagrama esquemático que ilustra un “posicionamiento” de dispositivos de servidor *front-end* en (o asignación de dispositivos de servidor *front-end* a) un espacio de identificador según una realización de la invención.

La figura 8 muestra un diagrama esquemático que ilustra un esquema de almacenamiento geográficamente redundante según una realización de la invención.

Descripción de las realizaciones de la invención

La figura 1 muestra un diagrama de bloques esquemático que ilustra una disposición a modo de ejemplo de un sistema según una realización de la invención.

El sistema comprende una pluralidad de servidores de un primer tipo de servidor que incluye un primer servidor y un segundo servidor (es decir, 10a y 10b). El sistema no se limita al número de dos servidores del primer tipo de servidor tal como se muestra en la figura 1 sino que puede comprender más de dos servidores del primer tipo de servidor. Los servidores del primer tipo de servidor constituyen el sistema *front-end*. El sistema puede comprender además una pluralidad de servidores de un segundo tipo de servidor que incluye un tercer servidor 20. El sistema no se limita al número de un servidor del segundo tipo de servidor tal como se muestra en la figura 1 sino que puede comprender más de un servidor del segundo tipo de servidor. Los servidores del segundo tipo de servidor constituyen el sistema *back-end*. La figura 1 también muestra un servidor 30 de aplicación que puede enviar un mensaje de consulta hacia el sistema. Ese mensaje de consulta se recibe por un servidor de la pluralidad de servidores del primer tipo de servidor, en el ejemplo ilustrado se recibe por el primer servidor 10a.

El primer servidor 10a comprende un transformador 11 y un procesador 12. El primer servidor 10a tiene acceso a una primera tabla 13 de búsqueda y a una respectiva segunda tabla 14 de búsqueda. Las tablas 13, 14 de búsqueda pueden estar almacenadas en una unidad de almacenamiento (no mostrada) del servidor 10a, tal como un disco duro, RAM o cualquier otra unidad de almacenamiento.

El primer servidor 10a puede comprender además una unidad 16 de interfaz (una primera unidad de interfaz) que proporciona una interfaz entre el primer servidor 10a y los otros servidores del primer tipo de servidor incluyendo el segundo servidor 10b. El primer servidor 10a puede tener una unidad 17 de interfaz adicional (una segunda unidad de interfaz) que proporciona una interfaz entre el primer servidor 10a y la pluralidad de servidores del segundo tipo de servidor incluyendo el tercer servidor 20, y una unidad 15 de interfaz (una tercera unidad de interfaz) que proporciona una interfaz entre el primer servidor 10a y servidores, tales como el servidor 30 de aplicación, que inician una consulta.

El segundo servidor 10b puede tener una configuración similar a la del servidor 10a. Concretamente, en caso de que la consulta se envíe del servidor 30 de aplicación al segundo servidor 10b, el segundo servidor 10b funciona como el primer servidor 10a puesto que es el “primer” servidor en el sistema de servidor *front-end* que recibe la consulta.

El transformador 11 transforma al menos parte de los datos contenidos en un mensaje de consulta, recibido a través de la tercera unidad 15 de interfaz del servidor 30 de aplicación, en un resultado de transformación.

El procesador 12 determina, basándose en el resultado de transformación y la primera tabla 13 de búsqueda, una ubicación de almacenamiento de una respectiva segunda tabla de búsqueda en uno de la pluralidad de servidores del primer tipo de servidor, accede a la respectiva segunda tabla 14 de búsqueda en el primer servidor 10a si el procesador 12 determina que la ubicación de almacenamiento de la respectiva segunda tabla de búsqueda está en el primer servidor 10a, y recupera un indicador que indica una ubicación de almacenamiento de los datos solicitados por el mensaje de consulta a partir de la respectiva segunda tabla 14 de búsqueda.

Basándose en el indicador recuperado, la segunda unidad 17 de interfaz puede extraer los datos solicitados por el mensaje de consulta desde un servidor de la pluralidad de servidores del segundo tipo de servidor, por ejemplo, el tercer servidor 20.

La primera unidad 16 de interfaz puede reenviar el resultado de transformación o el mensaje de consulta a otro servidor de la pluralidad de servidores del primer tipo de servidor, por ejemplo, al segundo servidor 10b, si el procesador 12 determina que la ubicación de almacenamiento de la respectiva segunda tabla de búsqueda no está en el primer servidor 10a sino en otro servidor del primer tipo de servidor que constituye el sistema *front-end*, por ejemplo, en el segundo servidor 10b.

Con referencia al segundo servidor 10b, en caso de que el mensaje de consulta se haya reenviado del primer servidor 10a al segundo servidor 10b, a través de la primera unidad 16 de interfaz, un transformador del segundo servidor 10b transforma al menos parte de los datos contenidos en el mensaje de consulta en un resultado de transformación.

Un procesador del segundo servidor 10b determina (es decir, concluye), basándose en el hecho de que recibe el mensaje de consulta o el resultado de transformación del primer servidor 10a, que la ubicación de almacenamiento de la respectiva segunda tabla de búsqueda está en el segundo servidor 10b, accede a la respectiva segunda tabla de búsqueda, y recupera un indicador que indica una ubicación de almacenamiento de los datos solicitados por el mensaje de consulta a partir de la respectiva segunda tabla de búsqueda. (Obsérvese que un mensaje de consulta reenviado desde un servidor *front-end* a otro servidor *front-end* puede distinguirse de ese modo de un mensaje de consulta enviado desde un servidor de aplicación a un servidor *front-end*, por ejemplo, mediante al menos un bit en el mensaje que indica el nodo respectivo o tipo de nodo que envío el mensaje al servidor *front-end* que recibe el

mensaje de consulta). Una primera unidad de interfaz del segundo servidor 10b puede enviar el indicador recuperado al primer servidor 10a, y la primera unidad 16 de interfaz puede recibir el indicador recuperado del segundo servidor 10b.

Basándose en el indicador recibido del segundo servidor 10b, la segunda unidad 17 de interfaz puede extraer los datos solicitados por el mensaje de consulta del tercer servidor 20.

Alternativamente, a través de su tercera unidad de interfaz, el segundo servidor 10b puede extraer los datos solicitados por el mensaje de consulta del tercer servidor 20 y reenviar los datos al primer servidor 10a. La primera unidad 16 de interfaz puede recibir los datos solicitados por el mensaje de consulta del segundo servidor 10b. Alternativamente, a través de su tercera unidad de interfaz, el segundo servidor 10b puede reenviar los datos al servidor 30 de aplicación.

Con referencia de nuevo al segundo servidor 10b, este servidor comprende una unidad de interfaz (primera unidad de interfaz, no mostrada) que puede recibir un resultado de transformación basándose en al menos parte de los datos contenidos en un mensaje de consulta, un procesador (no mostrado) puede concluir, basándose en la recepción del resultado de transformación, que una ubicación de almacenamiento de una respectiva segunda tabla de búsqueda está en el segundo servidor 10b, acceder a la respectiva segunda tabla de búsqueda, y recuperar un indicador que indica una ubicación de almacenamiento de los datos solicitados por el mensaje de consulta a partir de la respectiva segunda tabla de búsqueda.

La unidad de interfaz puede estar configurada para devolver el indicador recuperado del segundo servidor 10b a otro servidor desde el que se recibió el resultado de transformación o el mensaje de consulta, por ejemplo, el primer servidor 10a.

La segunda tabla de búsqueda puede comprender al menos un par de datos, que incluyen, para cada resultado de transformación, un puntero que indica una ubicación de los datos almacenados en un servidor del segundo tipo de servidor, en el que las segundas tablas de búsqueda de todos los servidores del primer tipo de servidor constituyen una tabla de búsqueda completa.

Las respectivas segundas tablas de búsqueda de cada servidor del primer tipo de servidor pueden ser iguales entre sí en cuanto a tamaño, o diferentes entre sí en cuanto a tamaño o bien en proporción a una respectiva capacidad de procesamiento de cada uno de los servidores o bien de tal manera que una carga de procesamiento para cada uno de los servidores del primer tipo de servidor sean iguales entre sí.

La primera tabla de búsqueda puede comprender al menos un par de datos, comprendiendo cada par de datos una dirección de un servidor de un primer tipo de servidor y un intervalo de valores definido por un valor de inicio y un valor de fin, en un espacio en el que se transforma la parte de los datos en el mensaje de consulta.

Los servidores del primer tipo de servidor pueden ser servidores *front-end* y la dirección puede ser una dirección IP de los respectivos servidores.

El transformador 11 puede mapear una cadena con un número entero natural.

El transformador 11 puede aplicar una transformada de *Hash*, y el espacio puede ser un espacio de *Hash* resultante de la transformada de *Hash*.

En este caso, las funciones de los servidores primero y segundo relevantes para entender los principios de la invención se describen usando unidades y/o bloques funcionales tal como se muestra en la figura 1. No debe interpretarse que la disposición de los bloques del primer servidor limita la invención, y las funciones pueden realizarse por un bloque (por ejemplo, con respecto a las unidades de interfaz) o dividirse adicionalmente en subbloques (por ejemplo, con respecto al procesador).

Las figuras 2A y 2B muestran un diagrama de flujo que ilustra un método de consulta de base de datos según una realización de la invención.

En la etapa S221, se recibe un mensaje de consulta en un primer servidor de una pluralidad de servidores de un primer tipo de servidor. En la etapa S222, al menos parte de los datos contenidos en el mensaje de consulta se transforma en un resultado de transformación. Por ejemplo, a la parte de los datos puede aplicarse un *hash* para dar un resultado de aplicación de *hash* (es decir, someterse a una transformada de *Hash*).

En la etapa S223, basándose en el resultado de transformación y en una primera tabla de búsqueda en el primer servidor, se determina una ubicación de almacenamiento de una respectiva segunda tabla de búsqueda en uno de la pluralidad de servidores del primer tipo. Por ejemplo, el resultado de aplicación de *hash* y la primera tabla de búsqueda pueden compararse en el primer servidor y puede determinarse la ubicación de un puntero de datos solicitados en el mensaje de consulta.

En la etapa S224 se decide si la ubicación de almacenamiento de la respectiva segunda tabla de búsqueda, por ejemplo, la ubicación del puntero de los datos solicitados, está en el primer servidor o no. Si es que SÍ en la etapa S224, es decir, la determinación en la etapa S223 arroja que la ubicación de almacenamiento de la respectiva segunda tabla de búsqueda está en el primer servidor, el primer servidor accede a su respectiva segunda tabla de búsqueda, y se recupera un indicador (por ejemplo, el puntero) que indica una ubicación de almacenamiento de los datos solicitados por el mensaje de consulta a partir de la respectiva segunda tabla de búsqueda en el primer servidor en la etapa S225.

En la etapa S226, puede enviarse una petición del primer servidor a un tercer servidor, es decir, un servidor de un segundo tipo de servidor, indicado por el puntero para extraer de ese modo los datos solicitados, y los datos extraídos pueden enviarse a un servidor que inicia el mensaje de consulta, por ejemplo, un servidor de aplicación, en la etapa S227.

Sin embargo, si es que NO en la etapa S224, es decir, la determinación en la etapa S223 arroja que la ubicación de almacenamiento de la respectiva segunda tabla de búsqueda está en un segundo servidor del primer tipo de servidor diferente del primer servidor, en la etapa S228 el mensaje de consulta se reenvía del primer servidor al segundo servidor.

En la etapa S229, en el segundo servidor al menos parte de los datos contenidos en el mensaje de consulta se transforman en el resultado de transformación. Por ejemplo, a la parte de los datos puede aplicarse un *hash* para obtener el resultado de aplicación de *hash*.

En la etapa S230, basándose en la recepción del mensaje de consulta en el segundo servidor que se origina del primer servidor, y basándose en el resultado de transformación obtenido partiendo del mismo en el segundo servidor, se concluye que la ubicación de almacenamiento de la respectiva segunda tabla de búsqueda está en el segundo servidor. El segundo servidor del primer tipo de servidor almacena la respectiva segunda tabla de búsqueda que incluye un indicador (puntero) que indica una ubicación de almacenamiento de los datos solicitados por el mensaje de consulta. Por ejemplo, basándose en el resultado de aplicación de *hash* la ubicación del puntero puede recuperarse a partir de la segunda tabla de búsqueda en el segundo servidor.

En la etapa S231, el indicador (puntero) que indica la ubicación de almacenamiento de los datos solicitados por el mensaje de consulta se determina en el segundo servidor a partir de la respectiva segunda tabla de búsqueda.

En la etapa S232 se decide en el segundo servidor si el puntero debe enviarse al primer servidor. Si es que SÍ en la etapa S232, el segundo servidor envía el puntero al primer servidor en la etapa S233, y el primer servidor puede realizar las etapas S226 y S227.

Sin embargo, si es que NO en la etapa S232, puede enviarse una petición del segundo servidor a un tercer servidor, es decir, un servidor de un segundo tipo de servidor, indicado por el puntero para extraer los datos solicitados en la etapa S234, y los datos extraídos pueden enviarse a un servidor que inicia el mensaje de consulta, por ejemplo, un servidor de aplicación, o al primer servidor en la etapa S235. En caso de que los datos solicitados se envíen al primer servidor en la etapa S235, el primer servidor puede realizar la etapa S227.

La decisión tomada en la etapa S232 puede basarse en parámetros configurados previamente mantenidos dentro del sistema. Asimismo, el destino alternativo para enviar los datos extraídos puede basarse en parámetros configurados previamente. Alternativamente, para una o ambas decisiones que van a tomarse, puede incluirse una respectiva orden, por ejemplo, en el mensaje de consulta.

La figura 3 muestra un diagrama de flujo que ilustra un método de consulta de base de datos según otra realización de la invención.

Las etapas S331 a S337 realizadas en un primer servidor corresponden a las etapas S221 a S227 en la figura 2. Debido a las etapas idénticas realizadas, en este caso se omite por tanto una descripción repetida de las etapas S331 a S333 y se hace referencia a la descripción de las correspondientes etapas S221 a S223. Lo mismo se aplica a las etapas S335 a S337 que corresponden a las etapas S225 a S227.

En la etapa S334 (que corresponde a la etapa S224) se decide si la ubicación de almacenamiento de la respectiva segunda tabla de búsqueda, por ejemplo, la ubicación del puntero de los datos solicitados, está o no en el primer servidor. Si es que SÍ en la etapa S334, es decir, la determinación en la etapa S333 arroja que la ubicación de almacenamiento de la respectiva segunda tabla de búsqueda está en el primer servidor, y se ejecutan las etapas S335 a S337.

Sin embargo, si es que NO en la etapa S334, es decir, la determinación en la etapa S333 arroja que la ubicación de almacenamiento de la respectiva segunda tabla de búsqueda está en otro servidor, es decir, un segundo servidor del primer tipo de servidor (siendo el segundo servidor diferente del primer servidor). Entonces, en la etapa S338 el resultado de transformación (resultado de aplicación de *hash*) se reenvía del primer servidor al segundo servidor.

En la etapa S339, se determina un indicador (puntero) que indica una ubicación de almacenamiento de los datos solicitados por el mensaje de consulta en el segundo servidor a partir de la respectiva segunda tabla de búsqueda en el segundo servidor, y en la etapa S340 se devuelve el puntero al primer servidor, y el primer servidor puede realizar las etapas S336 y S337.

En otras palabras, en la etapa S338, en el segundo servidor, del primer servidor de la pluralidad de servidores del primer tipo de servidor, se recibe un resultado de transformación basándose en al menos parte de los datos contenidos en un mensaje de consulta.

En la etapa S339, basándose en la recepción del resultado de transformación, el segundo servidor concluye que la ubicación de almacenamiento de la respectiva segunda tabla de búsqueda está en el segundo servidor, accede a la respectiva segunda tabla de búsqueda en el segundo servidor y recupera el indicador que indica la ubicación de almacenamiento de los datos solicitados por el mensaje de consulta a partir de la respectiva segunda tabla de búsqueda.

En la etapa S340, se devuelve el indicador recuperado del segundo servidor del primer tipo de servidor al primer servidor del primer tipo de servidor.

Tal como se describió anteriormente con relación a las figuras 2 y 3, respectivamente, se observa que cuando un segundo servidor FE recibe un mensaje de consulta (pero no un resultado de aplicación de *hash*), éste aplica un *hash* debido al formato de la segunda tabla de búsqueda. Pero, según una realización a modo de ejemplo, ya no comprueba la primera tabla de búsqueda. El formato de la segunda tabla de búsqueda es, por ejemplo, [*hash* (MSISDN), ID de servidor BE (dirección IP)]. Por tanto, reenviar únicamente el resultado de *hash* es suficiente para permitir que la búsqueda en la segunda tabla de búsqueda se mantenga en el segundo servidor *front-end*. Concretamente, en tal escenario a modo de ejemplo, el segundo servidor confía y concluye que tiene la clave en su segunda tabla de búsqueda, puesto que es responsable según la tabla de encaminamiento (la primera tabla de búsqueda) del primer servidor *front-end*. Si no puede encontrar la entrada en su segunda tabla de búsqueda (lo que sólo debería suceder si los nodos que almacenan realmente esa entrada han fallado), por consiguiente, esto significará una pérdida de la entrada. Lo anterior se basa en la suposición de que cada servidor tiene conocimiento acerca de todos los otros servidores y sus ID. Por tanto, sólo un (segundo) servidor *front-end* de este tipo se consultará internamente desde otro (primer) servidor *front-end* si se considera que es responsable de una entrada.

Según una realización de la invención, se usa un esquema de direccionamiento basado en *hash* para garantizar que la carga generada por las consultas de búsqueda (por ejemplo, peticiones de LDAP) se distribuya de manera uniforme y que pueda encontrarse cada entrada de búsqueda almacenada en un dispositivo de servidor *front-end* (que también se denomina nodo). Se aplica a sistemas de servidor *front-end*, es decir, sistemas en los que los nodos no contienen los datos finales. En su lugar, los servidores *front-end* mantienen sólo los punteros a los dispositivos de servidor *back-end* que son responsables de los datos “reales” o verdaderos.

La cantidad total de datos de búsqueda no se almacenan de manera completamente redundante en cada nodo *front-end* (lo que requeriría grandes cantidades de memoria en los nodos *front-end*), pero se divide y distribuye entre todos los dispositivos de servidor *front-end* que conforman todo el sistema, disminuyendo de ese modo la cantidad de memoria necesaria en cada dispositivo de servidor *front-end* a sólo una fracción de toda la memoria.

Según una realización de la invención, los datos se almacenan según el principio de tablas de *Hash* distribuidas (DHT), lo que significa que a todos los documentos así como las direcciones de los dispositivos *front-end* se les aplica un *hash* para dar un espacio de identificador. Los documentos se almacenan en el dispositivo *front-end* con el ID más bajo que aún es mayor que su propio ID (tal como se muestra en la figura 6). Como consecuencia, cada dispositivo de servidor *front-end* es responsable de un intervalo continuo del espacio de identificador y los documentos que se encuentran en este intervalo. Los dispositivos de servidor forman una estructura lógica (superposición) que se usa para identificar la ubicación de almacenamiento de un documento específico.

El mecanismo de funcionamiento básico de una función *hash* se muestra en la figura 6, tal como se usa, por ejemplo, en el presente caso en conexión con DHT (tablas de *Hash* distribuidas). Mapea los valores de entrada (en este ejemplo direcciones IP de servidores o nombres de documentos) con su codominio, que se usa como el espacio de identificador en el contexto de DHT. En esta figura, el codominio de la función *hash* es un espacio lineal, que oscila desde 1 hasta 2 m. Esto es en principio un valor igualmente fiable como el intervalo [1; 2^n-1[usado como un ejemplo en la presente invención. Otras variantes, tales como dos coordenadas, también son posibles.

Es importante observar que en esta figura, los servidores se mapean con el espacio de identificador a través de la misma función en contraste con esta invención. Sin embargo, se trata de una práctica usada y conocida en muchas DHT, en las que se usa un número mucho mayor de servidores y por tanto se necesita una generación de ID rápida, libre de colisión y económica. Debido a la aleatoriedad de muchas funciones *hash*, el posicionamiento de los servidores también es aleatorio cuando se usa este método, que en parte se remedia de nuevo por el número superior de servidores. Sin embargo, según esta invención, esto no es aplicable, motivo por el cual en esta invención los servidores no se colocan de manera aleatoria, sino siguiendo de manera determinística los algoritmos descritos.

La colocación de las entradas de búsqueda en la presente invención también se realiza basándose en una función *hash* aplicada a la clave de base de datos que se consulta, puesto que el simple número de entradas es suficiente para llevar a una distribución idéntica de entradas entre los servidores.

No obstante, otro principio mostrado en la figura es la colocación de documentos (archivos xml en el ejemplo de la figura, entradas de búsqueda desde la segunda tabla en la invención) en los servidores. En general, las superposiciones estructuradas tales como DHT obtienen su rendimiento de búsqueda comparablemente bueno a partir del hecho de que los documentos/objetos de datos tienen un servidor/nodo específico que es responsable de los mismos. Esta relación de responsabilidad puede derivarse normalmente de manera sencilla a partir de los valores *hash* (ID) de los objetos y servidores. Como consecuencia, si se busca un objeto, sólo el servidor que debe ser responsable de éste (si existe) debe ubicarse y consultarse. En el ejemplo de la figura, la regla para asignar servidores a objetos de datos es que un objeto se almacena en el servidor con el valor (ID) *hash* más bajo superior al valor *hash* del objeto. Esto es lo mismo en la presente invención (como también se representa en la figura 5). Por ejemplo, el documento con el valor *hash* 23 (objeto z) se almacena en el servidor cuyo ID sucede a este valor *hash*, en este caso el servidor x con valor/ID *hash* 42. Esto es el principio fundamental que debe haberse visualizado mediante la figura.

Según la invención, se proporcionan dos estructuras de datos, es decir, tablas de búsqueda primera y segunda, y dos procesos (dos etapas de búsqueda) con el fin de encaminar las consultas entrantes que alcanzan un dispositivo o servidor *front-end* arbitrario, por ejemplo, mediante un conmutador automático.

La primera estructura de datos (por ejemplo, primera tabla 13 de búsqueda) es una tabla de encaminamiento o "partición" pequeña. Contiene una tabla con una primera columna de ID de nodos de servidor *front-end* conocidos del sistema de base de datos distribuida, y una segunda columna que muestra de qué intervalos son responsables los nodos *front-end*. Según una realización de la invención, estos intervalos son partes de lo que se denomina habitualmente "espacio *hash*", por ejemplo todos los números desde 0 hasta 2^n. Los intervalos están separados de manera uniforme, de modo que más adelante, la carga en cada uno de los dispositivos de servidor *front-end* es la misma. Si el conjunto de dispositivos de servidor *front-end* es heterogéneo en capacidad, los intervalos también pueden dividirse según la capacidad individual de los dispositivos de servidor. Dependiendo del resultado de la función *hash*, es decir, una transformación de los datos de entrada obtenidos a partir de los datos contenidos en un mensaje de consulta de base de datos, puede derivarse inmediatamente a partir de la primera estructura de datos (primera tabla de búsqueda) el intervalo al que pertenece esta consulta, y por consiguiente, el dispositivo de servidor *front-end* que es responsable de esta consulta. Después de esto, se inicia un segundo proceso, que implica una segunda estructura de datos, es decir, la búsqueda en una respectiva segunda tabla de búsqueda.

La figura 4 muestra el mapeo entre secciones de datos, que se almacena en dispositivos *back-end*, y dispositivos *front-end* según una realización de la invención. *Los* nodos *front-end* que tienen los ID de ID1 a ID8 se asignan a intervalos en un espacio de ID de *hash* que incluye todos los números desde 1 hasta 2^{n-1}. La aplicación de un *hash* a un MSISDN como un ejemplo de datos contenidos en un mensaje de consulta recibido en uno arbitrario de los nodos *front-end* arroja un valor en el espacio de ID de *hash*. Por ejemplo, un ID4 de nodo *front-end* recibe el mensaje de consulta con el MSISDN +49(170)123456, que consulta un perfil de abonado. La aplicación de un *hash* a MSISDN arroja un valor en el espacio de ID que pertenece al intervalo del que es responsable el ID1 de nodo *front-end* (tal como se indica en la figura 4 mediante el triángulo negro ubicado dentro del espacio de ID de *hash*). Por tanto, el ID1 de nodo *front-end* almacena la parte de una tabla de búsqueda real que es necesaria para derivar un nodo *back-end* que contiene el perfil de abonado consultado para el MSISDN +49(170)123456.

Debe observarse que la figura 4 no muestra redundancia de responsabilidades.

Toda la segunda estructura de datos (es decir, formada de la pluralidad de respectivas segundas tablas 14 de búsqueda en los servidores *front-end*) es normalmente una tabla de búsqueda grande con millones de entradas, mientras que la tabla de encaminamiento (primera tabla de búsqueda) usada en el primer proceso sólo contiene tantas entradas como el número de servidores *front-end*. Según una realización de la invención, la segunda estructura de datos consiste en una columna que contiene claves, y una columna que contiene valores. Tal como se describió anteriormente, las claves pueden ser, por ejemplo, un MSISDN de *hash* y lo valores pueden ser los ID de nodo de servidores *back-end* (por ejemplo, HLR). Cuando se busca localmente, un nodo de servidor *front-end* aplicará *hash* al MSISDN y encontrará directamente el ID de nodo de servidor *back-end* correcto, correspondiente. Esto es posible ya que la estructura de tabla descrita se almacena en una estructura de datos de memoria "tabla *hash*" convencional, soportada por la mayoría de lenguajes de programación. El servidor *front-end* recuperará entonces los datos a partir del servidor *back-end* identificado y los devolverá al servidor *front-end* que recibe la consulta (el primer servidor *front-end*).

La figura 5 muestra un diagrama esquemático que ilustra el encaminamiento de consulta según una realización de la invención. Un sistema 100 de servidor *front-end* incluye nodos 101-110 de servidor *front-end*. En la etapa 1, el nodo 101 *front-end* recibe una consulta con una clave de búsqueda. En la etapa 2, el nodo 101 de servidor *front-end* aplica un *hash* a la clave de búsqueda y determina el almacenamiento de una entrada de búsqueda, es decir, qué nodo 101-110 de servidor *front-end* almacena la parte de la tabla de búsqueda que contiene el mapeo de la clave de

búsqueda con un nodo de servidor *back-end* o nodos de servidor de un sistema 200 de base de datos *back-end*. En caso de que el nodo 101 de servidor *front-end* determine que esta entrada de búsqueda está almacenada en un nodo 106 remoto, reenvía el resultado de aplicación de *hash* al nodo 106 remoto. En la etapa 3, el nodo 106 remoto busca el/los nodo(s) de servidor *back-end* y devuelve una indicación del/de los nodo(s) *back-end* buscado(s) (ubicación de datos solicitados por la consulta) al nodo 101 *front-end* en la etapa 4. En la etapa 5, el nodo 101 de servidor *front-end* busca una entrada en el/los nodo(s) de servidor *back-end* indicado(s) por el nodo 106 remoto, es decir, recupera los datos solicitados por la consulta.

La primera búsqueda en la primera tabla de búsqueda (tabla de partición) es muy rápida y sirve para descubrir el dispositivo de servidor *front-end* responsable. Sólo tal dispositivo *front-end* responsable puede realizar entonces la segunda búsqueda local en la respectiva segunda tabla de búsqueda, ya que sólo este dispositivo contiene los datos de tabla necesarios de la segunda estructura de datos, que finalmente lleva al dispositivo *back-end* correcto.

En comparación con los sistemas de base de datos distribuida convencionales, el sistema de la presente invención es mucho más fiable, más rápido y con calidad de portadora.

El sistema de la presente invención comprende múltiples nodos de autoorganización, descentralizados y cooperativos que pueden tener características de rendimiento heterogéneas.

La invención también permite un almacenamiento de datos geográficamente redundante a prueba de fallos que se describe a continuación y que puede lograr tiempos de respuesta y/o tiempos de contestación para devolver una contestación/respuesta a una petición que están dentro de ciertos límites predeterminados.

Tal como se describió anteriormente, el sistema de la presente invención requiere menos memoria (por ejemplo, RAM) a costa de un tráfico de red y carga de procesador de CPU ligeramente aumentados.

A continuación, se describe una realización de la invención que se refiere a un único “sitio” o dominio del sistema de búsqueda, en el que la cantidad total de datos de búsqueda se almacena en su totalidad, distribuidos entre los servidores *front-end* que constituyen el dominio. Más adelante se describe otra realización de la invención que se refiere a un sistema que consiste en varios sitios o dominios físicamente remotos tal como se muestra en la figura 8.

En el denominado sistema de búsqueda de un único sitio, según una realización de la invención, todos los dispositivos de servidor *front-end* que constituyen el sistema *front-end* están conectados lógicamente e integrados en una superposición. Esta superposición tiene una estructura en anillo tal como se muestra en la figura 5, que forma un espacio de ID usado para el direccionamiento y encaminamiento en esta superposición. Los dispositivos se conectan físicamente en una red de área local (LAN), con el fin de garantizar tiempos de transmisión cortos para mensajes entre los dispositivos de servidor (también denominados nodos).

Cada elemento de datos se coloca en la superposición aplicando un *hash* a la clave de búsqueda de la consulta de base de datos original (por ejemplo, el MSISDN en una base de datos de abonados). Este valor *hash* es al mismo tiempo la clave para el par <clave, valor> de la entrada de búsqueda correspondiente. Adicionalmente, cada dispositivo tiene su propia dirección de superposición, que corresponde a su posición en el espacio de ID. Las direcciones de todos los dispositivos determinan la cantidad de datos de búsqueda total que tiene que almacenar cada dispositivo, es decir, en caso de que se asigne un número de x direcciones diferentes de servidores *front-end*, cada uno de éstos almacena 1/x como un porcentaje de toda la segunda tabla de búsqueda, es decir, una respectiva segunda tabla de búsqueda en cada dispositivo de servidor *front-end* tiene un tamaño de 1/x de toda la cantidad de datos. Si se asignan 2x direcciones de servidor, toda la segunda tabla de búsqueda puede distribuirse y/o compartirse entre el doble de la cantidad de servidores *front-end*, de modo que cada uno de éstos necesita almacenar sólo la mitad de la cantidad de datos en comparación con el ejemplo anterior. Cada dispositivo 101-110 es responsable principalmente de las entradas de búsqueda que se colocan entre el ID de superposición del predecesor del dispositivo (es decir, el dispositivo con el ID más alto inferior al ID del dispositivo), y el ID de superposición del propio dispositivo (véanse las figura 6 y 7). Por ejemplo, el dispositivo 106 es responsable de las entradas de búsqueda que se colocan (en el espacio *hash*) entre el dispositivo 105 y el dispositivo 106. La figura 6 muestra la colocación del dispositivo y los datos en el espacio de ID (identificador). Cada objeto se almacena en un primer igual que sucede al valor *hash* de los objetos.

Existen diferentes opciones con respecto a la generación y asignación de ID de nodo. Un método es aplicar un “*hash*” a una dirección IP de un dispositivo. El resultado será una cadena de bits, por ejemplo, 128 ó 256 bits. Incluso si dos direcciones IP (protocolo de Internet) son muy similares (por ejemplo, 10.10.10.1 y 10.10.10.2), la salida *hash* es totalmente diferente y se distribuye a lo largo de una gran distancia en el espacio *hash* que comprende todos los números posibles.

Con la presente invención, cada nodo *front-end* sólo necesita almacenar una parte de la segunda tabla de búsqueda global que se constituye tras combinar todas las respectivas segundas tablas de búsqueda en los servidores *front-end*. Cuando llega una consulta a uno de los dispositivos de servidor *front-end*, por ejemplo, se extrae el MSISDN de una petición de LDAP y luego se aplica un *hash*. Basándose en el resultado de la función *hash*, el dispositivo *front-*

end puede determinar (basándose en la primera tabla de búsqueda) si los datos solicitados por la consulta están dentro de su propio intervalo válido del que es responsable este dispositivo *front-end*. Si el dispositivo *front-end* no es responsable de los datos solicitados, puede determinar dónde reenviar la consulta al dispositivo responsable de los datos solicitados, ya que todos los dispositivos *front-end* tienen una tabla de "encaminamiento" completa (es decir, la primera tabla 13 de búsqueda) disponible localmente.

Tal segundo dispositivo de servidor *front-end* se activa entonces por sí mismo y resuelve la consulta por sí mismo. De nuevo, puede aplicar un *hash* al MSISDN, pero esta vez sólo realiza una búsqueda de estructura de datos local en la segunda de las dos estructuras de datos, y esta vez recupera el ID del dispositivo *back-end* responsable que finalmente tiene el perfil de abonado completo y válido asociado al MSISDN. En realidad, el segundo dispositivo *front-end* no tiene que aplicar un *hash* al MSISDN de nuevo, ya que el valor *hash* puede reenviarse directamente mediante el primer dispositivo de servidor *front-end* al segundo servidor *front-end*. Sólo en caso de que el primer dispositivo de servidor *front-end* reenvíe la consulta original al segundo dispositivo *front-end*, el segundo dispositivo *front-end* tiene que aplicar un *hash* al MSISDN de nuevo. La consulta que se resuelve en el segundo dispositivo sólo puede fallar si se ha producido una pérdida de datos o los datos consultados no se almacenan en el sistema 200. Por tanto, no se necesita ninguna búsqueda adicional en la primera tabla de búsqueda del segundo servidor *front-end*.

Para proceder adicionalmente con la consulta, existen dos opciones.

Según una primera opción, el segundo dispositivo *front-end* consulta el propio dispositivo *back-end* identificado, recuperando el perfil u otros datos asociados con la clave de consulta, y devuelve el resultado completo al primer dispositivo *front-end* que entonces a su vez puede reenviar el perfil y otros datos de vuelta a una aplicación que puede haber enviado la consulta.

Según una segunda opción, el segundo dispositivo *front-end* sólo devuelve un puntero al dispositivo *back-end* identificado al primer dispositivo *front-end*, que luego consulta al dispositivo *back-end* y devuelve los resultados a la aplicación. Es importante observar, sin embargo, que la resolución completa de la consulta es un proceso de múltiples etapas tal como se muestra, por ejemplo, en las figuras 2, 3 y 5.

En primer lugar, se identifica un dispositivo *front-end* que tiene información acerca de qué dispositivo *back-end* contiene un conjunto de datos que está solicitando la consulta.

Luego, la consulta se reenvía al dispositivo *front-end* identificado. Este segundo dispositivo *front-end* busca localmente el dispositivo *back-end* que tendrá el conjunto de datos disponible que está solicitando la consulta. El segundo dispositivo *front-end* entonces o bien resuelve la consulta con el dispositivo *back-end* directamente, o bien, alternativamente, sólo envía el ID del dispositivo *back-end* al primer dispositivo *front-end* que entonces realizará la consulta final.

En cualquier caso, el resultado final del dispositivo *back-end* se devuelve a la aplicación que envía la consulta al primer dispositivo *front-end*.

Según otra realización, el dispositivo *back-end* envía el conjunto de datos que corresponde a la consulta recibida al primer dispositivo *front-end* en lugar de al segundo dispositivo *front-end*, de modo que el segundo dispositivo *front-end* pasa a no tener estado. Para este fin, tiene que añadirse una dirección IP del primer dispositivo *front-end* a la consulta durante la fase de encaminamiento al dispositivo *back-end*.

Según una realización adicional, el dispositivo *back-end* envía el conjunto de datos que corresponde a la consulta recibida directamente a la aplicación que emite la consulta, de modo que los dispositivos *front-end* primero y segundo pasan a no tener estado. Para este fin, en un escenario a modo de ejemplo de este tipo, tiene que añadirse una dirección IP de la aplicación a la consulta durante la fase de encaminamiento al dispositivo *back-end*.

Según todavía una realización adicional, los nodos de servidor *front-end* también pueden ubicarse en la misma plataforma y los mismos nodos que los servidores de aplicación, ahorrándose así el conjunto de dispositivos *front-end* dedicados.

Con el fin de introducir un equilibrio de carga en el sistema, cada dispositivo debe almacenar aproximadamente la misma cantidad de datos de búsqueda, o la cantidad de datos de búsqueda se equilibra según las condiciones del dispositivo. Esto garantiza que el consumo de memoria en cada nodo sea igual, permitiendo la instalación de menos memoria en todos los nodos, y que la misma cantidad de consultas tenga que resolverse por cada nodo. Para lograr esto, los dispositivos *front-end* se colocan equidistantes en el anillo de identificador (mapeado con el espacio de ID (*hash*)). Tal superposición se muestra, por ejemplo, en las figuras 4 y 7. Los dispositivos 901-904 de servidor *front-end* se colocan en un círculo 905 de identificador que se divide igualmente entre los dispositivos 901-904. Las entradas de tabla de búsqueda (es decir, la segunda tabla 14 de búsqueda) se almacenan en el dispositivo con el ID más pequeño superior a sus propios ID. Los ID de dispositivos adyacentes deben tener una distancia de intervalo de ID de 1/n para n dispositivos. Con el círculo 905 de identificador igualmente dividido, el espacio de identificador está

igualmente dividido entre los dispositivos 901-904 tal como se muestra en la figura 7.

Sin embargo, puesto que el sistema debe ser transparente a una aplicación externa que consulta la base de datos, cada dispositivo *front-end* debe poder recibir consultas para cualquiera de los conjuntos de datos en el sistema *back-end*, y por consiguiente tiene que poder resolver estas consultas. Puesto que un dispositivo de servidor *front-end* no contiene los segundos datos de búsqueda completos sino sólo una parte respectiva de los mismos, se introduce un mecanismo de encaminamiento interno con el fin de alcanzar el dispositivo remoto en el que se ubica la entrada de búsqueda para una consulta dada. Según el esquema de direccionamiento descrito anteriormente, cada dispositivo de servidor *front-end* almacena una tabla pequeña de encaminamiento, es decir, la primera tabla de búsqueda, que comprende todos los dispositivos *front-end* en el sistema, con sus ID de superposición como direcciones. Para una consulta de base de datos dada, un dispositivo *front-end* puede decidir qué otro dispositivo *front-end* es responsable de la entrada de búsqueda correspondiente aplicando un *hash* a la clave de búsqueda de base de datos y recuperando el dispositivo *front-end* con el ID más bajo superior a la clave *hash*.

Según una realización de la invención, se necesita como mucho un reenvío del mensaje de consulta o del resultado de transformación desde un primer servidor *front-end* que inicialmente recibe el mensaje de consulta a otro servidor *front-end* (segundo servidor *front-end*).

Un dispositivo *front-end* según la presente invención puede tener varios módulos de software que le permitan ofrecer la funcionalidad descrita anteriormente.

Según una realización, el dispositivo *front-end* comprende un módulo *hash* que tiene que aceptar todos los tipos de clave como entrada que pueden usarse como claves para la consulta de base de datos. Usa una función *hash* para mapear estas claves con el espacio de identificador, por ejemplo, $[0; 2^{160}[$. Un ejemplo de una función de este tipo es el algoritmo SHA-1. El módulo *hash* devuelve el valor *hash* de una clave usada como entrada.

Además, el dispositivo *front-end* puede comprender un módulo de encaminamiento que almacena todos los nodos que se sabe que participan en el sistema *front-end* en una tabla de encaminamiento pequeña. Devuelve el dispositivo *front-end* responsable de la entrada de búsqueda de una clave dada.

El dispositivo *front-end* puede comprender además un módulo de búsqueda que contiene las entradas de búsqueda almacenadas en el dispositivo *front-end* local en una estructura de datos. Permite leer, escribir, crear y borrar funciones en estas entradas así como la recuperación de bloques enteros de entradas, es decir, intervalos completos del espacio de ID.

Según una realización adicional de la invención, el concepto de la invención se aplica de manera recursiva, con el fin de mejorar la fiabilidad, ajuste a escala y tolerancia del sistema global, que puede propagarse por múltiples ubicaciones físicas (representando una “ubicación” un sitio o dominio como se mencionó anteriormente) tal como se muestra en la figura 8. En caso de que el contenido completo de la tabla de búsqueda de entrada deba almacenarse no sólo en una ubicación física, sino que se copie a varios sitios para aumentar la redundancia y protección a fallos, el sistema lógico descrito anteriormente puede extenderse con el fin de incluir todos los dispositivos *front-end* en todos los sitios. Con el fin de lograr esto, los ID descritos anteriormente se tratan como ID locales “ID_{local}”. A éstos, se añade un prefijo “ID_{Sitio}” que indica el sitio, dando como resultado un ID global = $ID_{Sitio}|ID_{local}$. Mientras que no sea necesario un tráfico entre sitios, los ID locales son suficientes para el proceso de encaminamiento dentro del sitio tal como se describió anteriormente. Si tiene que establecerse una conexión entre dos dispositivos de servidor *front-end* remotos, el ID global se usa para identificarlos. Esto significa que la tabla de encaminamiento (primera tabla de búsqueda) tiene que ampliarse para contener no sólo los dispositivos *front-end* en el sitio local, sino todos los dispositivos *front-end* en cada sitio. La figura 8 muestra la estructura lógica de tal red de superposición.

Si debe recuperarse un conjunto de datos especial a partir de un sitio remoto, puede encontrarse construyendo un ID global a partir del ID de sitio del sitio remoto y el ID local para ese conjunto de datos.

La presente invención puede aplicarse al registro de ubicación local, repositorio de abonados, sistemas de base de datos con tecnología de red de portadora y de CPS (almacenamiento de perfil común).

Para el fin de la presente invención tal como se describió anteriormente, debe observarse que

- las etapas del método que probablemente se implementen como partes de código de software y que se ejecutan usando un procesador en una de las entidades de servidor son código de software independiente y pueden especificarse usando cualquier lenguaje de programación desarrollado conocido o futuro;

- las etapas y/o dispositivos del método que probablemente se implementen como componentes de hardware en una de las entidades de servidor son hardware independiente y pueden implementarse usando cualquier tecnología de hardware desarrollada conocida o futura o cualquier híbrido de éstas, tal como MOS, CMOS, BiCMOS, ECL, TTL, etc., usando por ejemplo componentes ASIC o componentes DSP, como un ejemplo;

- en general, cualquier etapa del método es adecuada para implementarse como software o mediante hardware sin cambiar la idea de la presente invención;

- los dispositivos pueden implementarse como dispositivos individuales, pero esto no excluye que se implementen de un modo distribuido a lo largo de todo el sistema, mientras que se conserve la funcionalidad del dispositivo.

Debe entenderse que la descripción anterior es ilustrativa de la invención y que no debe interpretarse como limitativa de la invención. A los expertos en la técnica se les podrán ocurrir diversas modificaciones y aplicaciones sin apartarse del alcance de la invención tal como se define por las reivindicaciones adjuntas.

REIVINDICACIONES

1. Método, que comprende:

recibir (S221; S331) un mensaje de consulta en un primer servidor de una pluralidad de servidores de un primer tipo de servidor;

transformar (S222; S332) al menos parte de los datos contenidos en el mensaje de consulta en un resultado de transformación;

determinar (S223; S333), basándose en el resultado de transformación y en una primera tabla de búsqueda en dicho primer servidor, una ubicación de almacenamiento de una respectiva segunda tabla de búsqueda en uno de dicha pluralidad de servidores de dicho primer tipo,

acceder (S225, S228; S335, S338) a la respectiva segunda tabla de búsqueda en la ubicación de almacenamiento determinada de la respectiva segunda tabla de búsqueda en uno de dicha pluralidad de servidores de dicho primer tipo, y

recuperar (S225, S228; S335, S338) un indicador que indica una ubicación de almacenamiento de los datos solicitados por el mensaje de consulta a partir de la respectiva segunda tabla de búsqueda,

indicando la primera tabla de búsqueda, para cada resultado de transformación, cuál de dicha pluralidad de servidores de dicho primer tipo es responsable de los datos solicitados por el mensaje de consulta,

comprendiendo la respectiva segunda tabla de búsqueda en un servidor del primer tipo de servidor al menos un par de datos, que incluyen, para cada resultado de transformación, un puntero que indica una ubicación de los datos almacenados en un servidor de un segundo tipo de servidor, siendo la respectiva segunda tabla de búsqueda en un servidor del primer tipo de servidor, parte de una segunda tabla de búsqueda completa, y

siendo los servidores del primer tipo de servidor, servidores *front-end* que no almacenan los datos solicitados por el mensaje de consulta, y siendo los servidores del segundo tipo de servidor, servidores *back-end* que almacenan los datos solicitados por el mensaje de consulta.

2. Método según la reivindicación 1, que comprende

llevar a cabo el acceso al primer servidor (S225; S335), si dicha determinación arroja que la ubicación de almacenamiento de la respectiva segunda tabla de búsqueda está en dicho primer servidor.

3. Método según la reivindicación 1, que comprende

reenviar (S228; S338) el resultado de transformación o el mensaje de consulta del primer servidor a un segundo servidor del primer tipo de servidor diferente del primer servidor, si dicha determinación arroja que la ubicación de almacenamiento de la respectiva segunda tabla de búsqueda está en el segundo servidor.

4. Método según la reivindicación 3, que comprende

recibir en el primer servidor del primer tipo de servidor, el indicador recuperado del segundo servidor del primer tipo de servidor, tras el acceso a la segunda tabla de búsqueda en el segundo servidor basándose en el resultado de transformación o el mensaje de consulta reenviado.

5. Método según la reivindicación 3, que comprende

recibir en el primer servidor, los datos solicitados por el mensaje de consulta, del segundo servidor del primer tipo de servidor tras el acceso a un servidor de una pluralidad de servidores de un segundo tipo de servidor basándose en el indicador recuperado partiendo del mensaje de consulta.

6. Método, que comprende

recibir (S228; S338), en un segundo servidor de un primer tipo de servidor a partir de un primer servidor de una pluralidad de servidores del primer tipo de servidor, un mensaje de consulta o un resultado de transformación basándose en al menos parte de los datos contenidos en el mensaje de consulta;

concluir (S230), basándose en la recepción del mensaje de consulta o el resultado de transformación, que una ubicación de almacenamiento de una respectiva segunda tabla de búsqueda está en el segundo servidor,

acceder (S231; S339) a la segunda tabla de búsqueda en el segundo servidor de dicho primer tipo, y

recuperar (S231; S339) un indicador que indica una ubicación de almacenamiento de los datos solicitados por el mensaje de consulta a partir de la segunda tabla de búsqueda,

comprendiendo la respectiva segunda tabla de búsqueda en un servidor del primer tipo de servidor al menos un par de datos, que incluyen, para cada mensaje de consulta o resultado de transformación, un puntero que indica una ubicación de los datos almacenados en un servidor de un segundo tipo de servidor, siendo la respectiva segunda tabla de búsqueda en un servidor del primer tipo de servidor, parte de una segunda tabla de búsqueda completa, y

siendo los servidores del primer tipo de servidor, servidores *front-end* que no almacenan los datos solicitados por el mensaje de consulta, y siendo los servidores del segundo tipo de servidor, servidores *back-end* que almacenan los datos solicitados por el mensaje de consulta.

7. Método según la reivindicación 6, que comprende además

transformar (S229) al menos parte de los datos contenidos en el mensaje de consulta en un resultado de transformación, en caso de que se reciba el mensaje de consulta.

8. Método según la reivindicación 6, que comprende

devolver (S233; S340) el indicador recuperado del segundo servidor del primer tipo de servidor al primer servidor del primer tipo de servidor.

9. Método según la reivindicación 1, en el que las respectivas segundas tablas de búsqueda de cada servidor del primer tipo de servidor son iguales entre sí en cuanto al tamaño, o diferentes entre sí en cuanto al tamaño

en proporción a una respectiva capacidad de procesamiento de cada uno de dichos servidores, o

de tal manera que una carga de procesamiento para cada uno de los servidores del primer tipo de servidor sean iguales entre sí.

10. Método según la reivindicación 1, en el que la primera tabla de búsqueda comprende al menos un par de datos, comprendiendo cada par de datos

una dirección de un servidor del primer tipo de servidor y un intervalo de valores definido por un valor de inicio y un valor de fin, en un espacio en el que se transforma la parte de los datos en el mensaje de consulta.

11. Método según la reivindicación 10, en el que

los servidores del primer tipo de servidor son servidores *front-end* y dicha dirección es una dirección IP de los respectivos servidores.

12. Método según la reivindicación 1 ó 7, en el que dicha transformación comprende mapear una cadena con un número entero natural.

13. Método según la reivindicación 10, en el que dicho espacio es un espacio de *Hash* que resulta de la transformación mediante una transformada de *Hash*.

14. Servidor (10a), que comprende:

una unidad (15) de interfaz configurada para recibir un mensaje de consulta;

un transformador (11) configurado para transformar al menos parte de los datos contenidos en el mensaje de consulta en un resultado de transformación;

una primera tabla (13) de búsqueda que indica, para cada resultado de transformación, cuál de una pluralidad de servidores de un primer tipo es responsable de los datos solicitados por el mensaje de consulta;

una respectiva segunda tabla (14) de búsqueda que comprende al menos un par de datos, que incluyen, para cada resultado de transformación, un puntero que indica una ubicación de los datos almacenados en un servidor de un segundo tipo de servidor; y

un procesador (12) configurado para

determinar, basándose en el resultado de transformación y en una primera tabla de búsqueda, una ubicación de almacenamiento de una respectiva segunda tabla de búsqueda en uno de una pluralidad de servidores de un primer tipo de servidor,

acceder a la respectiva segunda tabla de búsqueda en la ubicación de almacenamiento determinada de la respectiva segunda tabla de búsqueda en uno de dicha pluralidad de servidores de dicho primer tipo, y

recuperar un indicador que indica una ubicación de almacenamiento de los datos solicitados por el mensaje de consulta a partir de la segunda tabla de búsqueda,

siendo la respectiva segunda tabla de búsqueda parte de una segunda tabla de búsqueda completa, y

siendo el servidor, un servidor *front-end* como un servidor del primer tipo de servidor que no almacena los datos solicitados por el mensaje de consulta, y siendo los servidores del segundo tipo de servidor, servidores *back-end* que almacenan los datos solicitados por el mensaje de consulta.

15. Servidor según la reivindicación 14, que comprende

una unidad (16) de interfaz configurada para reenviar el resultado de transformación o el mensaje de consulta a otro servidor de la pluralidad de servidores, si dicho procesador determina que la ubicación de almacenamiento de la respectiva segunda tabla de búsqueda no está en el servidor.

16. Servidor según la reivindicación 15, en el que

la unidad (16) de interfaz está configurada para recibir el indicador recuperado del otro servidor.

17. Servidor según la reivindicación 15, en el que

la unidad (16) de interfaz está configurada para recibir los datos solicitados por el mensaje de consulta del otro servidor.

18. Servidor según la reivindicación 14, que comprende

una unidad de interfaz configurada para enviar el indicador recuperado o los datos solicitados por el mensaje de consulta a otro servidor.

19. Servidor (10b), que comprende

una unidad de interfaz configurada para recibir, desde otro servidor de una pluralidad de servidores de un primer tipo de servidor, un mensaje de consulta o un resultado de transformación basándose en al menos parte de los datos contenidos en el mensaje de consulta;

una primera tabla de búsqueda que indica, para cada resultado de transformación, cuál de una pluralidad de servidores de un primer tipo es responsable de los datos solicitados por el mensaje de consulta; una respectiva segunda tabla de búsqueda que comprende al menos un par de datos, que incluyen, para cada resultado de transformación, un puntero que indica una ubicación de los datos almacenados en un servidor de un segundo tipo de servidor; y

un procesador configurad para

concluir, basándose en la recepción del mensaje de consulta o el resultado de transformación, que una ubicación de almacenamiento de una respectiva segunda tabla de búsqueda está en el servidor,

acceder a la respectiva segunda tabla de búsqueda en el servidor, y

recuperar un indicador que indica una ubicación de almacenamiento de los datos solicitados por el mensaje de consulta a partir de la segunda tabla de búsqueda,

siendo la respectiva segunda tabla de búsqueda parte de una segunda tabla de búsqueda completa, y

siendo el servidor, un servidor *front-end* como un servidor del primer tipo de servidor que no almacena los datos solicitados por el mensaje de consulta, y siendo los servidores del segundo tipo de servidor, servidores *back-end* que almacenan los datos solicitados por el mensaje de consulta.

20. Servidor según la reivindicación 19, que comprende además un transformador configurado para transformar al menos parte de los datos contenidos en el mensaje de consulta en un resultado de transformación, en caso de que se reciba el mensaje de consulta.

21. Servidor según la reivindicación 19, en el que

la unidad de interfaz está configurada para devolver el indicador recuperado o los datos consultados por el mensaje de consulta, del servidor a otro servidor desde el que se recibió el mensaje de consulta o resultado de transformación.

22. Servidor según la reivindicación 14 ó 19, en el que el servidor es de un primer tipo de servidor, y en el que la respectiva segunda tabla de búsqueda comprende al menos un par de datos, que incluyen, para cada resultado de transformación, un puntero que indica una ubicación de los datos almacenados en un servidor de un segundo tipo de servidor, en el que las respectivas segundas tablas de búsqueda de todos los servidores del primer tipo de servidor constituyen una tabla de búsqueda completa.

23. Servidor según la reivindicación 22, en el que las respectivas segundas tablas de búsqueda de cada servidor del primer tipo de servidor son iguales entre sí en cuanto a tamaño, o diferentes entre sí en cuanto a tamaño

en proporción a una respectiva capacidad de procesamiento de cada uno de dichos servidores o

de tal manera que una carga de procesamiento para cada uno de los servidores del primer tipo de servidor sean iguales entre sí.

24. Servidor según la reivindicación 14 ó 19, en el que la primera tabla de búsqueda comprende al menos un par de datos, comprendiendo el par de datos

una dirección de servidores de un primer tipo de servidor y un intervalo de valores definido por un valor de inicio y un valor de fin, en un espacio en el que se transforma la parte de los datos en el mensaje de consulta.

25. Servidor según la reivindicación 24, en el que

los servidores del primer tipo de servidor son servidores *front-end* y dicha dirección es una dirección IP de los respectivos servidores.

26. Servidor según la reivindicación 14 ó 20, en el que dicho transformador está configurado para mapear una cadena con un número entero natural.

27. Servidor según la reivindicación 24, en el que dicho transformador está configurado para aplicar una transformada de *Hash*, y dicho espacio es un espacio de *Hash* que resulta de dicha transformada de *Hash*.

28. Programa informático que comprende instrucciones que pueden implementarse mediante procesador para realizar el método según una cualquiera de las reivindicaciones 1 a 13.

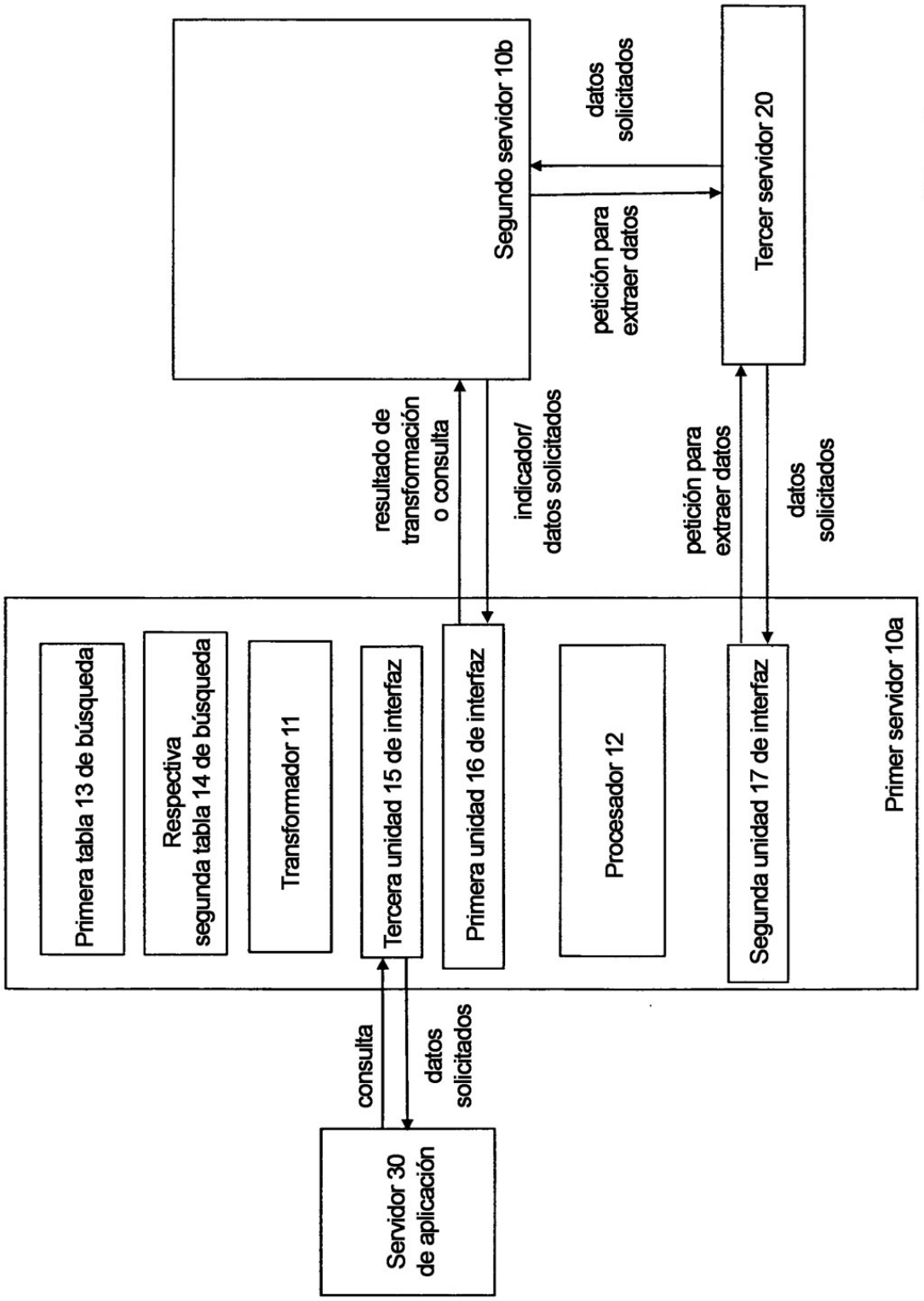
Segundo servidor 10b
datos solicitados
petición para extraer datos
Tercer servidor 20
resultado de transformación o consulta
indicador/ datos solicitados
petición para extraer datos
datos solicitados
Primera tabla 13 de búsqueda
Respectiva segunda tabla 14 de búsqueda
Transformador 11
Tercera unidad 15 de interfaz
Primera unidad 16 de interfaz
Procesador 12
Segunda unidad 17 de interfaz
Primer servidor 10a
consulta
datos solicitados
Servidor 30 de aplicación

Fig. 1

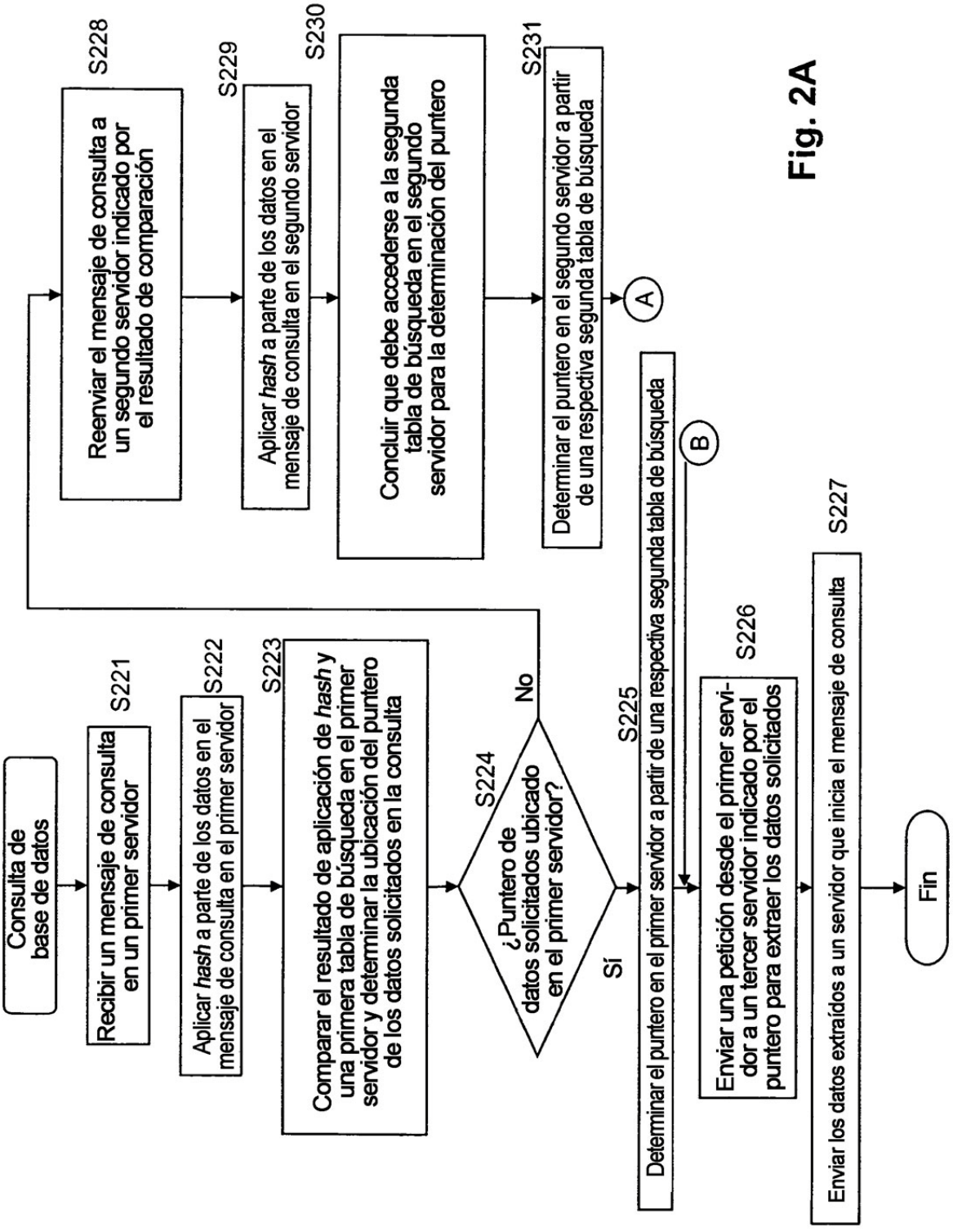
Consulta de base de datos
Recibir un mensaje de consulta en un primer servidor
S221
Aplicar hash a parte de los datos en el mensaje de consulta en el primer servidor
S222
Comparar el resultado de aplicación de hash y una primera tabla de búsqueda en el primer servidor y determinar la ubicación del puntero de los datos solicitados en la consulta
S223
¿Puntero de datos solicitados ubicado en el primer servidor?
S224
No
Sí
Reenviar el mensaje de consulta a un segundo servidor indicado por el resultado de comparación
S228
Aplicar hash a parte de los datos en el mensaje de consulta en el segundo servidor
S229
Concluir que debe accederse a la segunda tabla de búsqueda en el segundo servidor para la determinación del puntero
S230
Determinar el puntero en el segundo servidor a partir de una respectiva segunda tabla de búsqueda
S231
A
Determinar el puntero en el primer servidor a partir de una respectiva segunda tabla de búsqueda
S225
B
Enviar una petición desde el primer servidor a un tercer servidor indicado por el puntero para extraer los datos solicitados
S226
Enviar los datos extraídos a un servidor que inicia el mensaje de consulta
S227
Fin

Fig. 2A

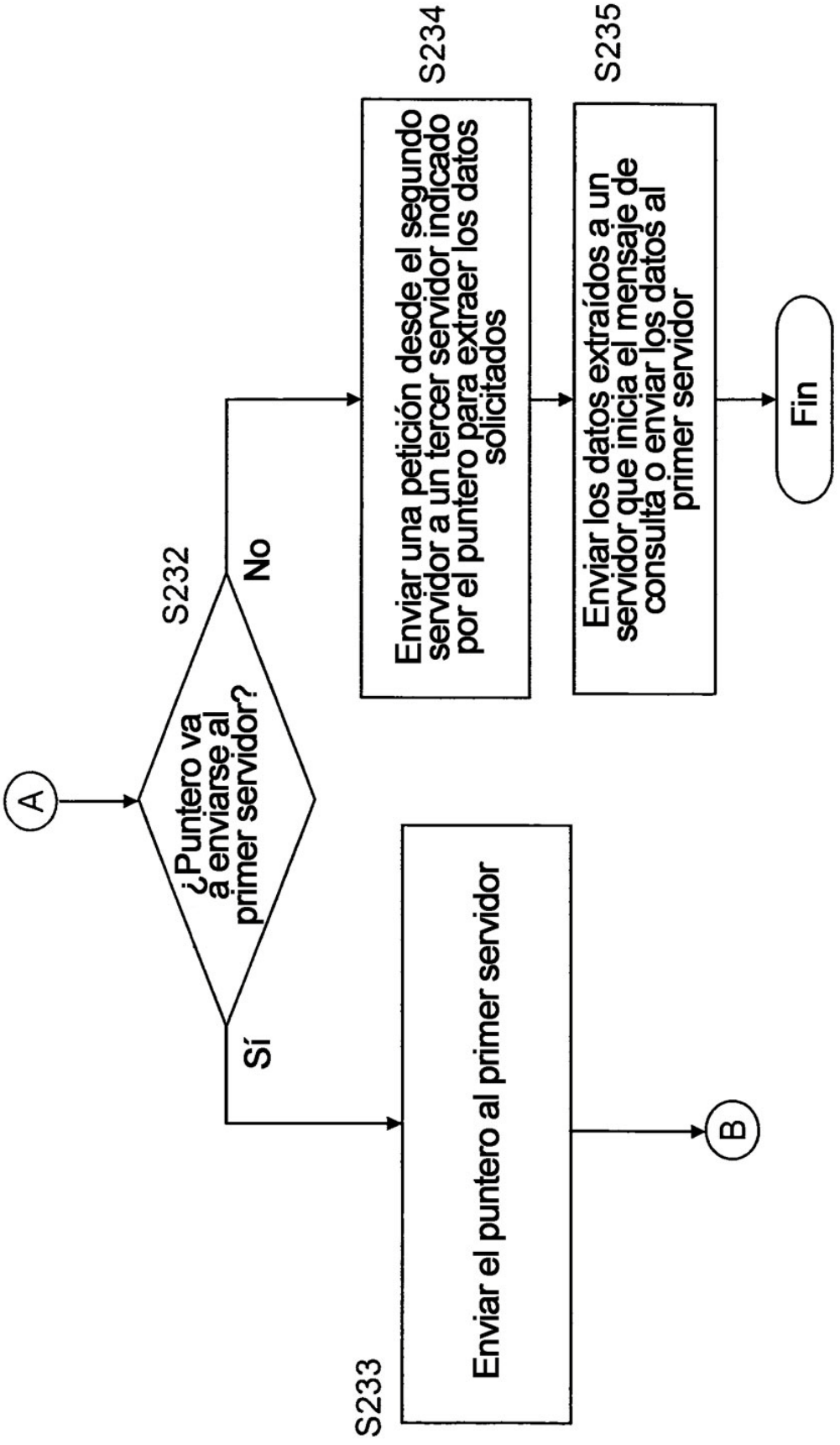
A
S232
¿Puntero va a enviarse al primer servidor?
No
Sí
S233
Enviar el puntero al primer servidor
B
S234
Enviar una petición desde el segundo servidor a un tercer servidor indicado por el puntero para extraer los datos solicitados
S235
Enviar los datos extraídos a un servidor que inicia el mensaje de consulta o enviar los datos al primer servidor
Fin

Fig. 2B

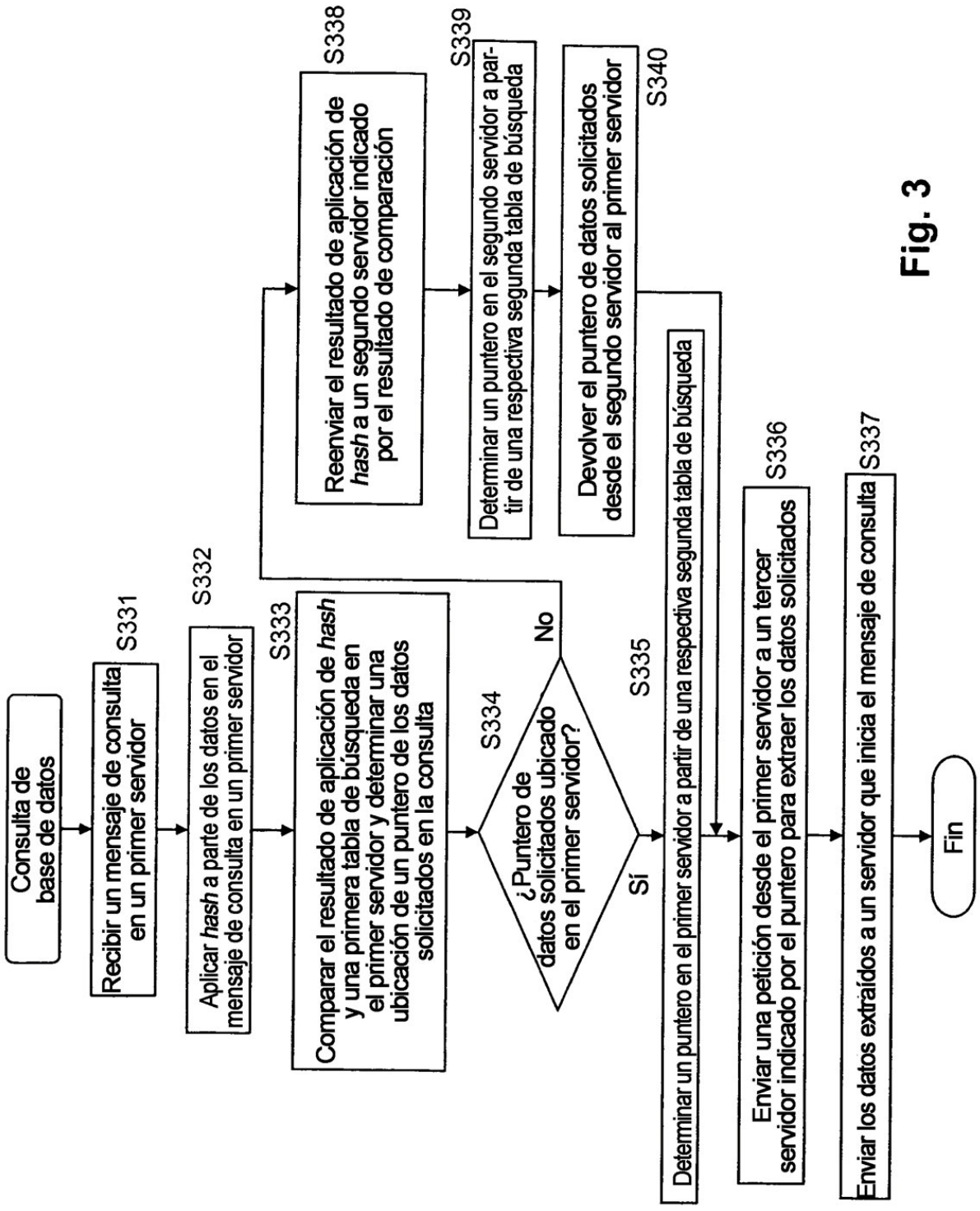
Consulta de base de datos
Recibir un mensaje de consulta en un primer servidor
S331
Aplicar *hash* a parte de los datos en el mensaje de consulta en un primer servidor
S332
Comparar el resultado de aplicación de *hash* y una primera tabla de búsqueda en el primer servidor y determinar una ubicación de un puntero de los datos solicitados en la consulta
S333
¿Puntero de datos solicitados ubicado en el primer servidor?
S334
No
Reenviar el resultado de aplicación de *hash* a un segundo servidor indicado por el resultado de comparación
S338
Determinar un puntero en el segundo servidor a partir de una respectiva segunda tabla de búsqueda
S339
Devolver el puntero de datos solicitados desde el segundo servidor al primer servidor
S340
Sí
S335
Determinar un puntero en el primer servidor a partir de una respectiva segunda tabla de búsqueda
Enviar una petición desde el primer servidor a un tercer servidor indicado por el puntero para extraer los datos solicitados
S336
Enviar los datos extraídos a un servidor que inicia el mensaje de consulta
S337
Fin

Fig. 3

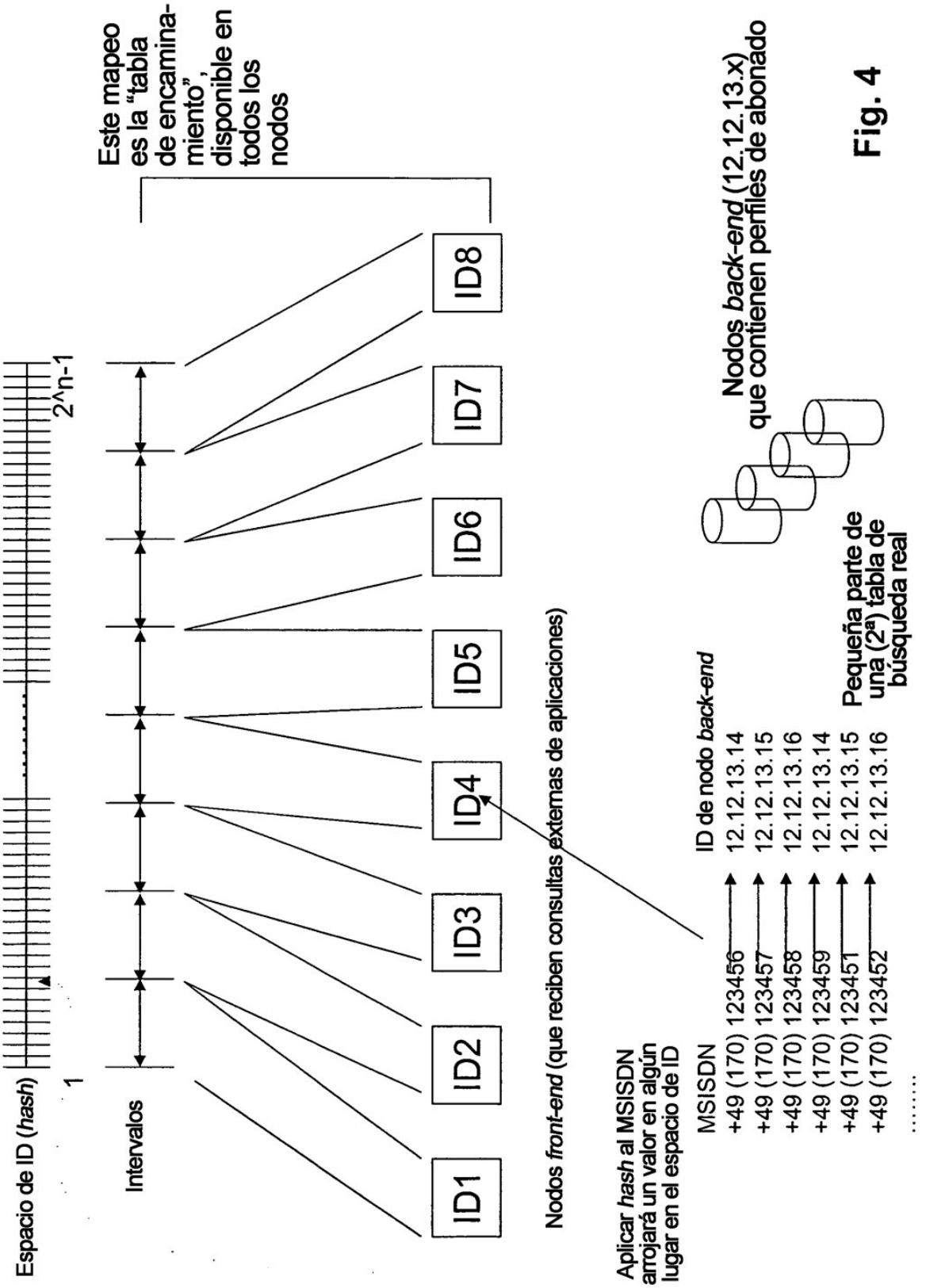
Este mapeo es la "tabla de encamina-miento", disponible en todos los nodos
Espacio de ID (hash)
1
2^n-1
Intervalos
ID1
ID2
ID3
ID4
ID5
ID6
ID7
ID8
Nodos front-end (que reciben consultas externas de aplicaciones)
Aplicar hash al MSISDN arrojará un valor en algún lugar en el espacio de ID
MSISDN
+49 (170) 123456
+49 (170) 123457
+49 (170) 123458
+49 (170) 123459
+49 (170) 123451
+49 (170) 123452
.....
ID de nodo back-end
12.12.13.14
12.12.13.15
12.12.13.16
12.12.13.14
12.12.13.15
12.12.13.16
Pequeña parte de una (2ª) tabla de búsqueda real
Nodos back-end (12.12.13.x) que contienen perfiles de abonado

Fig. 4

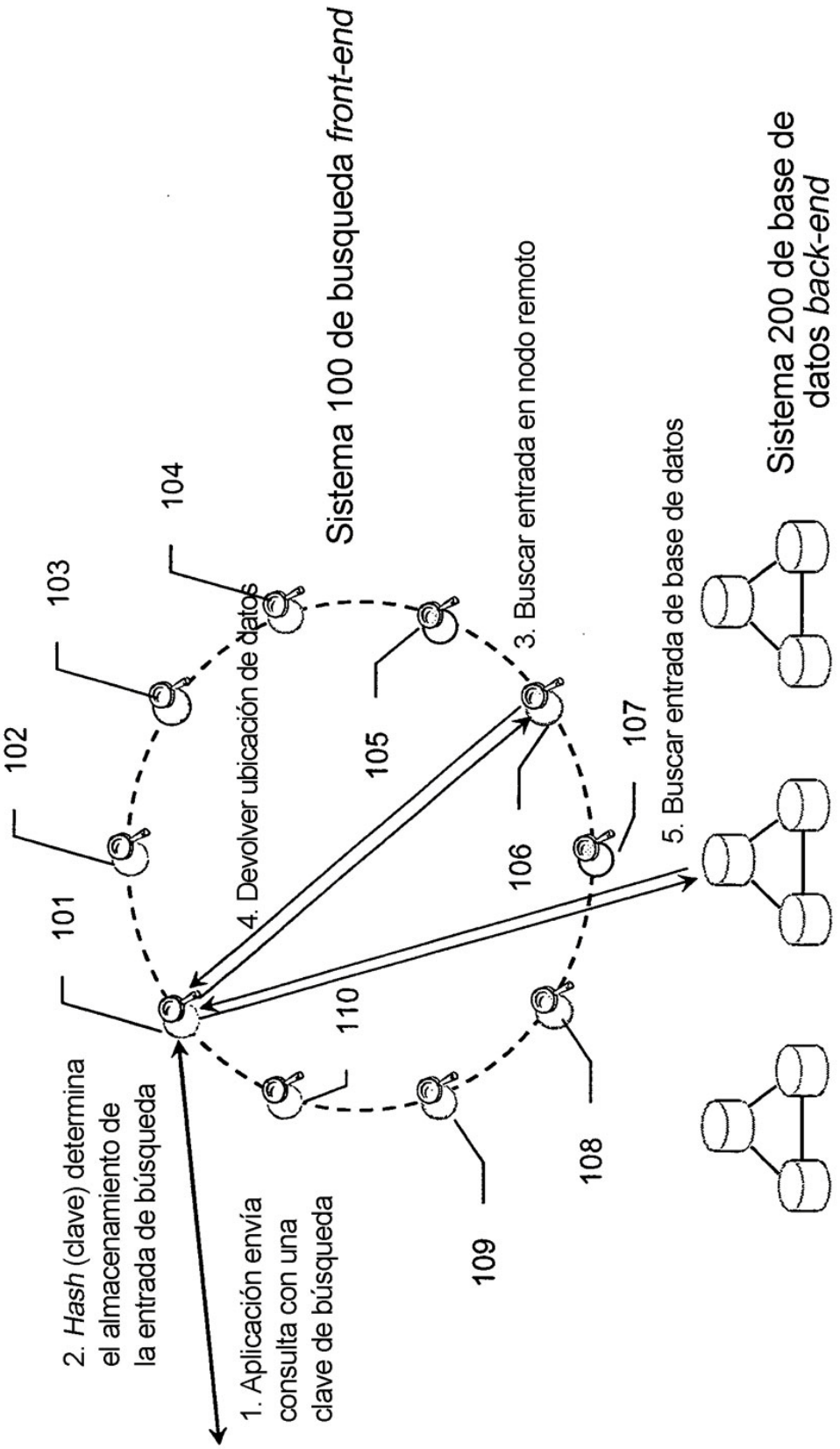
Sistema 100 de busqueda *front-end*
Sistema 200 de base de datos *back-end*
1. Aplicación envía consulta con una clave de búsqueda
2. *Hash* (clave) determina el almacenamiento de la entrada de búsqueda
3. Buscar entrada en nodo remoto
4. Devolver ubicación de datos
5. Buscar entrada de base de datos
101
102
103
104
105
106
107
108
109
110

Fig. 5

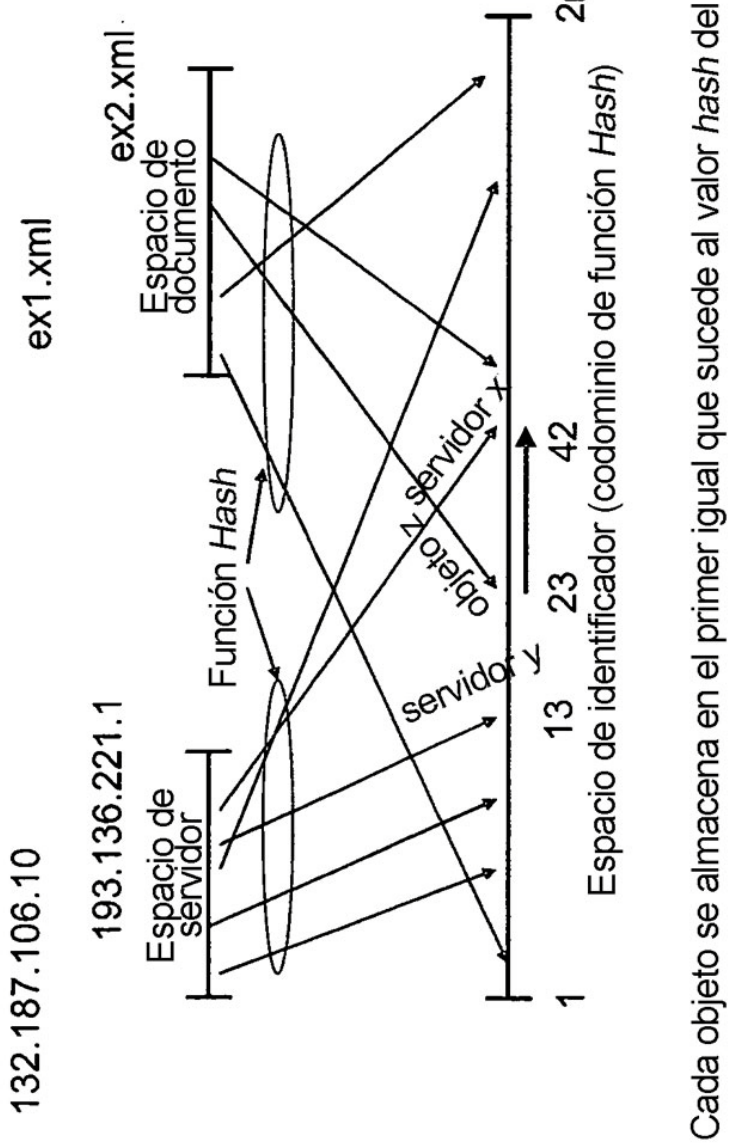
132.187.106.10
ex1.xml
193.136.221.1
Espacio de servidor
Función Hash
ex2.xml
Espacio de documento
objeto x
objeto z
servidor y
servidor x
1
13
23
42
2m
Espacio de identificador (codominio de función Hash)
Cada objeto se almacena en el primer igual que sucede al valor hash del objeto

Fig. 6

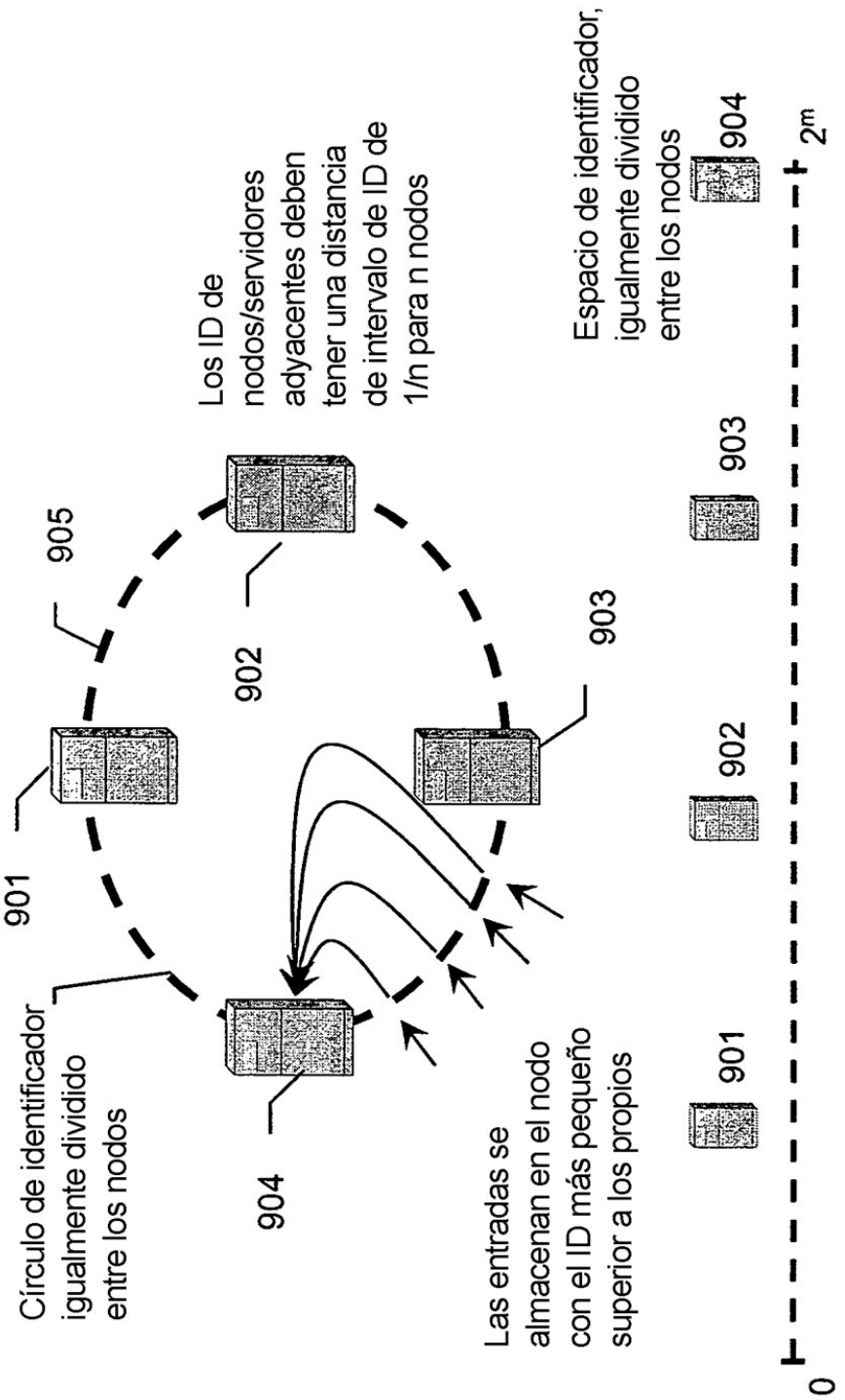
Círculo de identificador igualmente dividido entre los nodos
Las entradas se almacenan en el nodo con el ID más pequeño superior a los propios
Los ID de nodos/servidores adyacentes deben tener una distancia de intervalo de ID de 1/n para n nodos
Espacio de identificador, igualmente dividido entre los nodos
901
902
903
904
905
0
2^m

Fig. 7

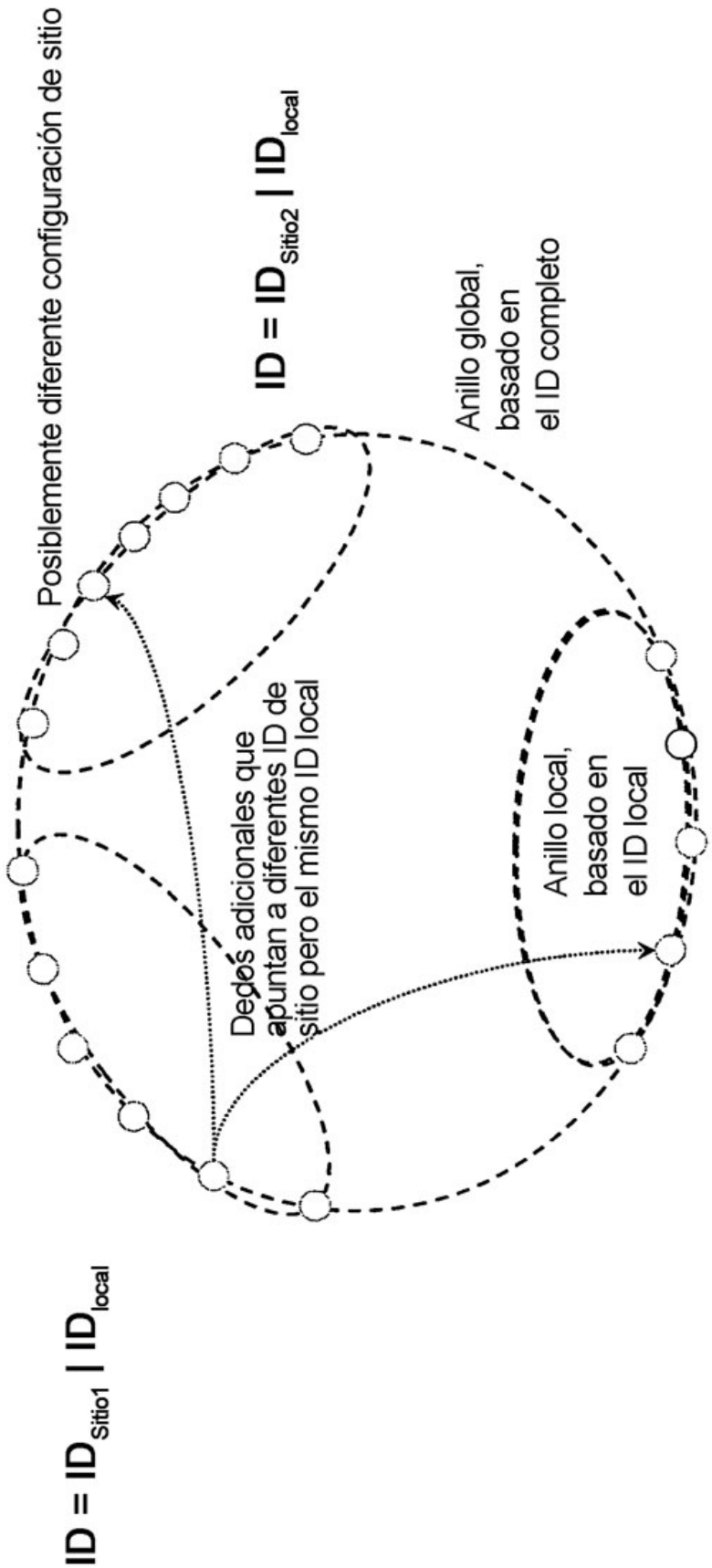
Posiblemente diferente configuración de sitio
ID = ID_Sitio2 | ID_local
Anillo global, basado en el ID completo
ID = ID_Sitio3 | ID_local
Dedos adicionales que apuntan a diferentes ID de sitio pero el mismo ID local
Anillo local, basado en el ID local
ID = ID_Sitio1 | ID_local
ID = ID_Sitio | ID_local

Fig. 8