(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 788 299**

(51) Int. CI.:

G06F 16/23 (2009.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **13.07.2011 PCT/US2011/043794**

(87) Fecha y número de publicación internacional: **19.01.2012 WO12009397**

(96) Fecha de presentación y número de la solicitud europea: **13.07.2011 E 11807426 (9)**

(97) Fecha y número de publicación de la concesión europea: **18.03.2020 EP 2593881**

(54) Título: **Compartición y resolución de conflictos de cambios de datos en un sistema de bases de datos multimaestro**

(30) Prioridad:

15.07.2010 US 836801

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
21.10.2020

(73) Titular/es:

PALANTIR TECHNOLOGIES, INC. (100.0%)
100 Hamilton Avenue, Suite 300
Palo Alto, California 94301, US

(72) Inventor/es:

GARROD, JOHN KENNETH;
CARRINO, JOHN ANTONIO;
BRAINARD, KATHERINE;
SCOTT, JACOB y
CHANG, ALLEN

(74) Agente/Representante:

ISERN JARA, Jorge

DESCRIPCIÓN

Compartición y resolución de conflictos de cambios de datos en un sistema de bases de datos multimaestro

Campo técnico

La presente descripción generalmente se refiere a sistemas informáticos distribuidos y, en particular, a la compartición y resolución de los conflictos de los cambios de datos en un sistema de bases de datos multimaestro.

Antecedentes

En un sistema de bases de datos multimaestro, los datos se almacenan en un grupo de bases de datos, se pueden realizar cambios de datos en cualquier miembro del grupo, y los cambios de datos realizados en un miembro se propagan al resto del grupo. Los sistemas de bases de datos multimaestro normalmente emplean o bien un esquema de replicación "síncrono" o bien uno "asíncrono" para propagar un cambio realizado en una base de datos al resto de las bases de datos del grupo.

En la replicación multimaestro síncrona, cada cambio se aplica a todas las bases de datos del grupo inmediatamente o a ninguna de las bases de datos si una o más de las bases de datos del grupo no pueden aceptar el cambio. Por ejemplo, una de las bases de datos puede no estar en línea o disponible. La replicación multimaestro síncrona generalmente se logra utilizando un protocolo de asignación de dos fases.

Por el contrario, en la replicación multimaestro "asíncrona", un cambio realizado en una base de datos es inmediatamente aceptado por la base de datos, pero la propagación del cambio a otras bases de datos en el grupo puede diferirse. Debido a que la propagación de los cambios puede diferirse, si una o más de las bases de datos del grupo no están disponibles, las bases de datos disponibles aún pueden aceptar cambios, poniendo en cola los cambios localmente hasta que puedan propagarse. Por esta razón, los sistemas de bases de datos multimaestro que emplean una estrategia de replicación asíncrona se consideran más disponibles que los sistemas de bases de datos multimaestro que emplean una estrategia de replicación síncrona. Sin embargo, la replicación asíncrona aumenta la posibilidad de conflictos que se producen como resultado de cambios concurrentes en la base de datos.

Un conflicto puede surgir en un sistema de bases de datos multimaestro cuando los mismos datos se cambian en dos bases de datos diferentes antes de que cualquiera de esos cambios pueda propagarse al otro. Por ejemplo, supóngase que en la base de datos A los datos que representan el color de ojos de una persona en particular se cambian a "marrón", y después de ese cambio, pero antes de que ese cambio pueda propagarse a la base de datos B, los datos en la base de datos B que representan el color de ojos de una persona en particular se cambian a "verde". Sin información adicional, no está claro qué cambio es el cambio "correcto" que deberían adoptar todas las bases de datos del sistema.

Los sistemas de bases de datos multimaestro que emplean un esquema de replicación asíncrona proporcionan normalmente mecanismos para "resolver conflictos". Según se usa en el presente documento, el término "resolver el conflicto" se refiere generalmente a la detección y resolución de un conflicto tal que una resolución del conflicto es en última instancia adoptada por todas las bases de datos del sistema. En algunos casos, el sistema de bases de datos multimaestro puede ser capaz de resolver el conflicto automáticamente sin requerir la intervención del usuario. En otros casos, se puede requerir la intervención del usuario para determinar cuál de los cambios concurrentes se debería adoptar.

En los sistemas de bases de datos multimaestro que emplean replicación asíncrona, cuándo se detectan los conflictos tiene un efecto enorme en la integridad de los datos de la base de datos. Por ejemplo, algunos sistemas de bases de datos pueden admitir la "resolución de objetos". La resolución de objetos involucra a un usuario o a un proceso informático automatizado que determina que dos o más objetos de datos distintos realmente representan la misma entidad del mundo real e invoca una función del sistema de bases de datos para que los objetos de datos distintos se resuelvan en un único objeto de datos. Por ejemplo, supóngase que hay dos objetos de datos distintos, uno con un valor de propiedad de nombre de "John Smith", el otro con un valor de propiedad de nombre de "J. S.". Un usuario puede decidir que estos dos objetos de datos representan a la misma persona en el mundo real. En consecuencia, en un sistema de bases de datos que admite la resolución de objetos, el usuario puede invocar una función del sistema de bases de datos para que los dos objetos de datos distintos se resuelvan en un único objeto de datos que tenga un valor de propiedad de nombre "John Smith" o "J. S." según lo seleccionado por el usuario que resuelve los objetos juntándolos.

En los sistemas de bases de datos multimaestro que emplean replicación asíncrona, sería deseable detectar como un conflicto cambios concurrentes que incluyen un cambio en la resolución del objeto. Por ejemplo, supóngase que en la base de datos A, el usuario 1 cambió la propiedad del color del cabello de un objeto de datos que representa a una persona llamada "J. S." de "castaño" a "rubio". Además, supóngase que antes de que el cambio de color del cabello realizado por el usuario 1 pueda propagarse desde la base de datos A a la base de datos B, el usuario 2 cambia la base de datos B resolviendo el objeto de datos que representa "J. S." juntándolo con otro objeto de datos

que representa a una persona llamada "John Smith". Sería deseable que el sistema de bases de datos multimaestro detecte estos dos cambios concurrentes como un conflicto ya que el usuario 2 pudiera no haber decidido resolver "J. S." y "John Smith" juntándolos si el usuario 2 hubiera sabido que el color del cabello de John Smith fue cambiado por el usuario 1. De manera similar, el usuario 1 pudiera no haber decidido cambiar el color del cabello de "J. S." si el usuario 1 hubiera sabido que el usuario 2 resolvió "J. S." y "John Smith" juntándolos.

Entonces, lo que se necesita es un sistema de bases de datos multimaestro que emplee replicación asíncrona que detecte conflictos resultantes de cambios concurrentes de una manera que esté en línea con las expectativas del usuario y que maneje la resolución de conflictos y la propagación de tales cambios de manera apropiada. Las realizaciones de la presente invención satisfacen estas y otras necesidades.

Los planteamientos descritos en esta sección son planteamientos que podrían seguirse, pero no necesariamente planteamientos que se hayan concebido o seguido previamente. Por lo tanto, a menos que se indique lo contrario, no se debe suponer que ninguno de los planteamientos descritos en esta sección cualifica como técnica anterior simplemente en virtud de su inclusión en esta sección.

Documento "Detection of mutual inconsistency in distributed system", por Stott Parker D. et al, Transacciones IEEE en Ingeniería de Software, Centro de Servicio IEEE, Los Alamitos, CA, EE. UU., Vol. SE-09, n° 3, 1 de mayo de 1983, páginas 240-247, XP000654801, ISSN: 0098-5589, describe cómo los vectores de versión pueden usarse para rastrear actualizaciones de objetos de archivo en un sistema de archivos distribuido y su aplicación en una replicación positiva.

Breve descripción de los dibujos

La presente invención se ilustra a modo de ejemplo, y no a modo de limitación, en las figuras de los dibujos adjuntos y en los que los números de referencia similares se refieren a elementos similares y en los que:

La FIG. 1 ilustra un sistema de bases de datos multimaestro para su uso en el intercambio y la resolución de conflictos de cambios de datos entre una pluralidad de sitios de replicación conforme a una realización de la invención.

La FIG. 2 ilustra un modelo de datos conceptual centrado en objetos conforme a una realización de la invención.

La FIG. 3 ilustra un método para compartir un cambio de datos a un objeto de datos en un sistema de bases de datos multimaestro usando vectores de versión por objeto, conforme a una realización de la invención.

La FIG. 4 ilustra un método para detectar y resolver un conflicto que involucra cambios concurrentes en un objeto de datos usando vectores de versión por objeto, conforme a una realización de la invención.

La FIG. 5 ilustra un ejemplo de detección y resolución de un conflicto que involucra cambios concurrentes a un objeto de datos usando vectores de versión por objeto conforme a una realización de la invención.

La FIG. 6 ilustra un ejemplo de intercambio de cambios de datos usando vectores de versión por conjunto de vínculos conforme a una realización de la invención.

La FIG. 7 ilustra un sistema informático con el que se puede implementar una realización.

Descripción detallada

Introducción

Con referencia a las figuras, a continuación se describirán realizaciones ejemplares de la invención. Las realizaciones ejemplares se describen principalmente con referencia a diagramas de bloques o a diagramas de flujo. En cuanto a los diagramas de flujo, cada bloque dentro de los diagramas de flujo representa tanto un paso del método como un elemento del aparato para realizar el paso del método. Dependiendo de la implementación, el elemento del aparato correspondiente puede configurarse en hardware, software, firmware o en combinaciones de los mismos.

Además, en la siguiente descripción, a efectos de explicación, se exponen numerosos detalles específicos para proporcionar una comprensión exhaustiva de la presente invención. Sin embargo, será evidente que la presente invención se puede emplear sin estos detalles específicos. En otros casos, los diagramas de bloques incluyen estructuras y dispositivos bien conocidos para evitar complicar innecesariamente la presente invención.

Resumen

Según una o más realizaciones de la presente invención, un sistema de bases de datos multimaestro y un método informático proporcionan la compartición y la resolución de conflictos de cambios de datos entre una pluralidad de sitios de replicación.

En una realización particular, los cambios de datos en sitios en los objetos de datos son rastreados por cada sitio en función de los objetos de datos usando vectores de versión por objetos de datos. El método incluye un primer dispositivo informático en un primer sitio que realiza un cambio en un objeto de datos. El primer dispositivo informático comparte el cambio en el objeto de datos con uno o más sitios. Un segundo dispositivo informático en un segundo sitio recibe una actualización que refleja el cambio en el objeto de datos realizado por el primer dispositivo informático en el primer sitio. La actualización incluye una identificación del objeto de datos, datos que reflejan el cambio en el objeto de datos y un vector de versión del objeto de datos en el primer sitio. El segundo dispositivo informático obtiene un vector de versión del objeto de datos en el segundo sitio y compara el vector de versión del objeto de datos en el primer sitio con el vector de versión del objeto de datos en el segundo sitio para determinar si los dos vectores de versión son idénticos, están ordenados o son concurrentes. En función de esta comparación, el segundo sitio intenta resolver automáticamente el conflicto entre las dos versiones del objeto de datos si, de acuerdo con sus vectores de versión, son concurrentes, o incorpora automáticamente la actualización recibida en la copia del objeto de datos del segundo sitio si, de acuerdo con sus vectores de versión, la versión del objeto de datos en el segundo sitio está ordenada antes de la versión recibida en la actualización.

En otra realización particular, los cambios de datos en los sitios en vínculos que conectan dos objetos de datos se rastrean en base al conjunto de vínculos usando vectores de versión por conjunto de vínculos. El método incluye un primer dispositivo informático en un primer sitio que realiza un cambio en un conjunto de vínculos que conectan dos objetos de datos. El primer dispositivo informático comparte el cambio en el conjunto de vínculos con uno o más sitios. Un segundo dispositivo informático en un segundo sitio recibe una actualización que refleja el cambio en el conjunto de vínculos realizado por el primer dispositivo informático en el primer sitio. La actualización incluye una identificación del conjunto de vínculos y un vector de versión del conjunto de vínculos en el primer sitio. El segundo dispositivo informático obtiene un vector de versión del conjunto de vínculos en el segundo sitio y compara el vector de versión del conjunto de vínculos en el primer sitio con el vector de versión del conjunto de vínculos en el segundo sitio para determinar si los dos vectores de versión son idénticos, están ordenados o son concurrentes. En función de esta comparación, el segundo sitio intenta resolver automáticamente el conflicto entre las dos versiones del conjunto de vínculos si, de acuerdo con sus vectores de versión, son concurrentes, o incorpora automáticamente la actualización recibida en la copia del segundo sitio del conjunto de vínculos si, de acuerdo con sus vectores de versión, la versión del conjunto de vínculos en el segundo sitio está ordenada antes de la versión recibida en la actualización.

En otra realización particular, los vectores de versión por objeto se usan para detectar un conflicto resultante de cambios concurrentes en dos o más sitios en los que al menos uno de los cambios concurrentes incluye un cambio en la resolución del objeto. El método incluye un primer dispositivo informático en un primer sitio de la pluralidad de sitios que resuelven dos o más objetos de datos juntándolos mediante un elemento de resolución de objetos de un sistema de bases de datos o de una aplicación de base de datos. El primer dispositivo informático comparte el cambio en la resolución con uno o más sitios de la pluralidad de sitios. Un segundo dispositivo informático recibe una actualización que refleja el cambio en la resolución realizado por el primer dispositivo informático en el primer sitio. La actualización incluye una identificación de cada uno de los dos o más objetos de datos que se resolvieron juntándolos y, para cada uno de los dos o más objetos de datos, un vector de versión del objeto de datos en el primer sitio. El segundo dispositivo informático obtiene, para cada uno de los dos o más objetos de datos, un vector de versión del objeto de datos en el segundo sitio. El segundo dispositivo informático compara, para cada uno de los dos o más objetos de datos, el vector de versión del objeto de datos en el primer sitio con el vector de versión del objeto de datos en el segundo sitio para determinar si las dos versiones son idénticas, están ordenadas, o son concurrente. En respuesta al segundo dispositivo informático que determina que el vector de versión de al menos un objeto de datos de los dos o más objetos de datos en el primer sitio es concurrente con el vector de versión de al menos un objeto de datos en el segundo sitio, el segundo dispositivo informático determina que el cambio en la resolución realizado por el primer dispositivo informático en el primer sitio entra en conflicto con la versión de al menos un objeto de datos en el segundo sitio.

Otras realizaciones incluyen, sin limitación, un medio no transitorio legible por ordenador que incluye instrucciones ejecutables por procesador que permiten que una unidad de procesamiento implemente uno o más aspectos de los métodos descritos, así como un sistema configurado para implementar uno o más aspectos de los métodos descritos.

Sistema de bases de datos multimaestro con motor de resolución de conflictos

La FIG. 1 ilustra un sistema 100 de base de datos multimaestro para su uso en la compartición y resolución de conflictos de cambios de datos entre una pluralidad de sitios de replicación conforme a una realización de la invención. En una realización, los sitios 101, 102 y 103 están unidos a través de una o más redes de datos tales como Internet, una o más redes de área amplia (WAN), una o más redes de área local (LAN), uno o más buses de comunicación de red o alguna combinación de los mismos. No es necesario que exista una red de datos extremada

o continuamente disponible entre los sitios de replicación y la(s) red(es) de datos que conecta(n) dos sitios cualesquiera puede(n) estar solo disponibles periódicamente. En otra realización, uno o más de los sitios no están conectados a ningún otro sitio en el sistema y los datos se transportan a y desde estos sitios manualmente utilizando medios portátiles o un dispositivo de medios portátil tal como un disco compacto (CD), un disco digital versátil (DVD), un dispositivo flash de bus de serie universal (USB), etc.

Cada sitio 101, 102 y 103 puede comprender uno o más dispositivos informáticos en red, tales como uno o más ordenadores de estaciones de trabajo, ordenadores de servidores, ordenadores portátiles, dispositivos informáticos móviles, o combinaciones de los mismos conectados entre sí a través de una o más redes de datos. Además, aunque solo se muestran tres sitios en la FIG. 1, el sistema 100 de base de datos multimaestro puede comprender muchos cientos o incluso muchos miles de sitios distribuidos geográficamente.

Según una realización, cada sitio 101, 102 y 103 tiene copias 111, 112 y 113 del mismo cuerpo de datos. El cuerpo de datos puede ser, por ejemplo, una o más tablas en una base de datos relacional. Sin embargo, las realizaciones de la invención no están limitadas a bases de datos relacionales y puede usarse cualquier tipo de base de datos capaz de admitir el modelo de datos conceptuales descrito en el presente documento. Los ejemplos no limitantes de tipos de bases de datos capaces de admitir el modelo de datos conceptual descrito en el presente documento incluyen las bases de datos relacionales, las bases de datos jerárquicas y las bases de datos orientadas a objetos.

Con respecto a ese cuerpo de datos particular, el sitio 101 puede configurarse para propagar de manera asíncrona al sitio 102 los cambios realizados en la copia 111, y propagar de manera asíncrona al sitio 103 los cambios realizados en la copia 111. De manera similar, el sitio 102 puede configurarse para propagar de manera asíncrona al sitio 101 los cambios realizados en la copia 112, y propagar de manera asíncrona al sitio 103 los cambios realizados en la copia 212. El sitio 103 puede configurarse para propagar de manera asíncrona a ambos sitios 101 y 102 los cambios realizados en la copia 113. Sin embargo, no es necesario que cada sitio deba configurarse para propagar a cualquier otro sitio los cambios realizados en su copia del cuerpo de datos. En otras palabras, no se requiere una topología de sitio multimaestro de malla completa para implementar las realizaciones de la invención y se pueden usar topologías multimaestro de malla parcial o en cascada.

Como el sistema 100 emplea un esquema de replicación asíncrona, cada copia 111, 112 y 113 del cuerpo de datos es de manera general consistente con las otras copias. Es decir, cada copia puede diferir de vez en cuando, de modo que en cualquier momento dado la vista de una copia del cuerpo de datos puede ser diferente de la vista de otra copia del cuerpo de datos. En ausencia de nuevos cambios, se espera que las copias se vuelvan consistentes entre sí. Por lo tanto, además de ser de manera general coherentes entre sí, también se puede decir que las copias 112, 112, 113, etc., son en última instancia consistentes.

Cada sitio 101, 102 y 103 tiene una lógica 120 de resolución de conflictos para recibir cambios remotos al cuerpo de datos desde otros sitios, detectar conflictos, resolver conflictos detectados bien de forma automática o bien con asistencia del usuario, y compartir cambios locales en el cuerpo de datos con otros sitios. La lógica 120 de resolución de conflictos puede implementarse como uno o más programas de software de ordenador, una o más lógicas programables de campo, lógica cableada o una combinación de las mismas. En una realización, la lógica 120 de resolución de conflictos es un componente de software de un sistema de gestión de base de datos tal como los disponibles comercialmente de Oracle Corporation de Redwood Shores, California y de Microsoft Corporation de Redmond Washington. En otra realización, la lógica 120 de resolución de conflictos es un componente de software de una aplicación basada en red, basada en servidor o de escritorio que utiliza un sistema de gestión de base de datos para realizar las técnicas de resolución de conflictos descritas en el presente documento. En otra realización más, la lógica 120 de resolución de conflictos se implementa en parte por una aplicación basada en red, basada en servidor o de escritorio y en parte por un sistema de gestión de base de datos.

Según se usa en el presente documento, el término "cambio", a menos que sea evidente de otra manera por el texto circundante, se refiere a una adición, edición o eliminación en una copia del cuerpo de datos en un sitio. Un cambio puede ser iniciado por un usuario o un proceso informático. Además, un cambio también puede iniciarse mediante la lógica 120 de resolución de conflictos en respuesta a la recepción de la notificación de un cambio anterior realizado en un sitio diferente del sitio que recibe la notificación.

Según se usa en el presente documento, el término "actualización", a menos que sea evidente de otra manera por el texto circundante, se refiere a la información sobre un cambio que se envía desde el sitio que realizó el cambio a otro sitio. Cada cambio puede dar lugar a que se reciba una actualización en todos los otros sitios para que los otros sitios puedan incorporar el cambio en sus respectivas copias del cuerpo de datos. La recepción de una actualización en un sitio puede generar un conflicto con la copia del cuerpo de datos del sitio receptor. Las técnicas implementadas por la lógica 120 de resolución de conflictos para detectar y resolver conflictos en diversos escenarios se describen con mayor detalle a continuación.

Modelo de datos centrado en objetos

En una realización, el cuerpo de datos, del cual cada sitio 101, 102 y 103 guarda una copia, está estructurado conceptualmente conforme a un modelo de datos centrado en objetos. Debe entenderse que este modelo de datos conceptual es independiente de cualquier modelo de datos de base de datos particular que pueda usarse para almacenar una copia del cuerpo de datos en un sitio. Por ejemplo, cada objeto del modelo de datos conceptual puede corresponder a una o más filas en una base de datos relacional o una entrada en la base de datos de protocolo ligero de acceso a directorios (LDAP).

La FIG. 2 ilustra un modelo 200 de datos conceptuales centrado en objetos según una realización. El modelo 200 se centra en la noción de un objeto 201 de datos. En el nivel más alto de abstracción, el objeto 201 de datos es un contenedor de información que representa cosas del mundo. Por ejemplo, el objeto 201 de datos puede representar una entidad tal como una persona, un lugar, una organización u otro sustantivo. El objeto 201 de datos puede representar un acontecimiento que ocurre en un punto en el tiempo o durante un tiempo. El objeto 201 de datos puede representar un documento u otra fuente de datos no estructurada, tal como un mensaje de correo electrónico, un informe de noticias o un artículo o documento escrito. Como mínimo, cada objeto 201 de datos está asociado con un identificador único que identifica de manera única el objeto de datos dentro del sistema 100. Cada objeto 201 de datos también puede tener un tipo (por ejemplo, Persona, Acontecimiento o Documento) y un nombre para mostrar que puede ser el valor de una propiedad particular del objeto de datos.

Cada objeto 201 de datos puede tener una o más propiedades 203. Las propiedades 203 son atributos del objeto 201 de datos que representan elementos de datos individuales. Como mínimo, cada propiedad 203 de un objeto 201 de datos tiene un tipo y un valor. Los diferentes tipos de objetos de datos pueden tener diferentes tipos de propiedades. Por ejemplo, un objeto de datos Persona podría tener una propiedad Color de ojos y un objeto Acontecimiento podría tener una propiedad Fecha. En una realización, el conjunto de tipos de objetos de datos y el conjunto de tipos de propiedades para cada tipo de objeto de datos admitido por el sistema 100 se definen de acuerdo con una ontología predefinida o definida por el usuario o con otra estructura jerárquica de conocimiento mediante subcategorización de tipos de objetos y tipos de propiedades según sus cualidades relevantes y/o cognitivas. Además, el modelo 200 de datos puede admitir multiplicidad de propiedades. En particular, se puede permitir que un objeto 201 de datos tenga más de una propiedad 203 del mismo tipo. Por ejemplo, un objeto de datos Persona puede tener múltiples propiedades de Dirección o múltiples propiedades de Nombre.

Cada vínculo 202 representa una conexión entre dos objetos 201 de datos. En una realización, la conexión es a través de una relación, de un acontecimiento o mediante propiedades coincidentes.

Una conexión de relación puede ser asimétrica o simétrica. Por ejemplo, el objeto de datos A Persona puede estar conectado al objeto de datos B Persona por una relación de Hijo de (donde el objeto de datos B Persona tiene una relación asimétrica de Padre de con el objeto de datos A Persona), una relación simétrica de Parentesco con el objeto de datos C Persona, y una relación asimétrica de Miembro de con el objeto de datos X Organización. El tipo de relación entre dos objetos de datos puede variar según los tipos de objetos de datos. Por ejemplo, el objeto de datos A Persona puede tener una relación de Aparecer en con el objeto de datos Y Documento o tener una relación de Participar en con el objeto de datos E Acontecimiento.

Como ejemplo de una conexión de acontecimiento, dos objetos de datos de Persona pueden estar conectados por un objeto de datos de Vuelo de línea aérea que representa un vuelo de una línea aérea particular si viajaron juntos en ese vuelo, o por un objeto de datos de Reunión que representa una reunión particular si ambos asistieron a esa reunión. En una realización, cuando dos objetos de datos están conectados por un acontecimiento, también están conectados por relaciones, en las que cada objeto tiene una relación específica con el acontecimiento, tal como, por ejemplo, una relación de Aparece en.

Como ejemplo de una conexión de propiedades coincidentes, dos objetos de datos Persona que representan un hermano y una hermana pueden tener ambos una propiedad Dirección que indica dónde viven. Si el hermano y la hermana viven en la misma casa, entonces sus propiedades de Dirección probablemente contengan información similar, si no idéntica. En una realización, se puede establecer un vínculo entre dos objetos de datos basándose en propiedades similares o coincidentes de los objetos de datos.

Los anteriores son solamente algunos ejemplos de los tipos de conexiones que pueden representarse mediante un vínculo y se pueden representar otros tipos de conexiones. Por lo tanto, debe entenderse que las realizaciones de la invención no están limitadas a ningún tipo particular de conexiones entre objetos de datos. Por ejemplo, un documento puede contener dos entidades etiquetadas diferentes. Un vínculo entre dos objetos de datos puede representar una conexión entre estas dos entidades a través de su aparición conjunta dentro del mismo documento.

Cada objeto 201 de datos puede tener múltiples vínculos con otro objeto 201 de datos para formar un conjunto 204 de vínculos. Por ejemplo, dos objetos de datos de Persona que representan un esposo y una esposa podrían estar vinculados a través de una relación de Cónyuge de, una propiedad coincidente (Dirección) y un acontecimiento (Boda).

En una realización, el modelo 200 de datos admite la resolución de objetos. Como se mencionó anteriormente, la resolución de objetos incluye un usuario o un proceso informático automatizado que determina que dos o más objetos 201 de datos distintos realmente representan la misma entidad del mundo real e invoca una función del sistema 100 en un sitio 101, 102, 103, etc. para que los objetos de datos distintos aparezcan ante los usuarios del sistema 100 como si fueran un único objeto de datos. En una realización, cuando un objeto 201 de datos se resuelve juntándolo con otro objeto 201 de datos, las propiedades y vínculos de un objeto de datos se copian en el otro objeto de datos y luego se eliminan del objeto de datos desde el que se copiaron. Sin embargo, ambos objetos de datos aún son retenidos por el sistema. Además de facilitar la capacidad de separar problemas de objetos de datos que previamente se resolvieron juntándolos, retener los objetos de datos después de resolverlos juntándolos facilita la detección y resolución de conflictos según se describe con mayor detalle a continuación.

Vectores de versión por objeto de datos

Un vector de versión es un mecanismo conocido para rastrear cambios en sistemas distribuidos. Sin embargo, los vectores de versión se emplean normalmente en función del sitio. Es decir, con implementaciones típicas de vectores de versión en sistemas distribuidos, cada sitio usa un único vector de versión para rastrear todos los cambios realizados en la copia de la base de datos guardada por ese sitio.

Según una realización de la invención, para rastrear y resolver los conflictos de los cambios en el cuerpo de datos, cada sitio 101, 102, 103, etc. guarda vectores de versión en función de cada objeto de datos. Al hacerlo, los conflictos que involucran cambios en las propiedades de los objetos de datos y los conflictos que involucran cambios en la resolución de los objetos pueden detectarse y resolverse adecuadamente como se explica con mayor detalle a continuación.

En una realización, cada sitio guarda un vector de versión para cada objeto de datos gestionado por el sistema. Por lo tanto, para un sistema que tiene m sitios que gestionan n objetos de datos, cada sitio guardará n vectores de versión para un total de $m * n$ vectores de versión guardados por todos los m sitios. Cada vector de versión puede contener hasta m elementos, uno para cada uno de los m sitios. Cada elemento de un vector de versión tiene un valor que representa un reloj lógico para el objeto de datos asociado en el sitio correspondiente al elemento. En una realización práctica, para conservar el espacio de almacenamiento de datos, los datos guardados en un sitio que representa un vector de versión pueden no representar todos los m elementos, sino algún subconjunto de los m elementos. Por ejemplo, los elementos de un vector de versión que tienen un valor predeterminado pueden no estar representados.

Cada sitio tiene, en cada vector de versión que guarda el sitio, su propio valor de reloj lógico como uno de los elementos. Este valor de reloj lógico representa la versión del objeto de datos asociado en el sitio que guarda el vector de versión. Cada otro elemento en el vector de versión representa la mejor estimación del sitio en función de las actualizaciones que el sitio ha recibido de la versión del objeto de datos asociado en el sitio correspondiente al otro elemento.

En una realización, cada elemento de un vector de versión se establece en algún valor inicial (por ejemplo, cero). Cuando un sitio cambia una o más propiedades de un objeto de datos en una transacción de base de datos frente a la copia del sitio del cuerpo de datos, el sitio incrementa su propio reloj lógico en el vector de versión asociado con el objeto de datos en un valor fijo (por ejemplo, uno). Al compartir el cambio con otros sitios como una actualización, el sitio incluye en la actualización datos que representan el cambio al objeto de datos y datos que representan el vector de versión del sitio para el objeto de datos después del incremento. Un sitio que recibe la actualización puede comparar el vector de versión en la actualización con su propio vector de versión para el objeto de datos para determinar si la versión del objeto de datos en el sitio receptor y la versión del objeto de datos en la actualización son (1) idénticos, están (2) ordenados o son (3) concurrentes.

Se pueden usar técnicas conocidas para comparar dos vectores de versión para determinar si las dos versiones son idénticas, están ordenadas o son concurrentes. En una realización, comparar dos vectores de versión incluye comparar cada elemento en un vector de versión con el elemento correspondiente en el otro vector de versión. La correspondencia entre elementos se determina en función del sitio al que corresponden los elementos. En particular, el elemento para un sitio en un vector de versión se compara con el elemento para el mismo sitio en el otro vector de versión. Dos versiones son idénticas si cada elemento en un vector de versión es igual al elemento correspondiente en el otro vector de versión. Las dos versiones están ordenadas si una versión "sucedió antes" que la otra. El vector A de versión sucedió antes del vector B de versión si cada elemento en el vector B de versión es mayor o igual que el elemento correspondiente en el vector A de versión y al menos un elemento en el vector B de versión es mayor que el elemento correspondiente en el vector A de versión. De manera similar, el vector B de versión ocurrió antes del vector A de versión si cada elemento en el vector A de versión es mayor o igual al elemento correspondiente en el vector B de versión y al menos un elemento en el vector A de versión es mayor que el elemento correspondiente en el vector B de versión. Dos versiones son concurrentes si no son idénticas ni están ordenadas.

Compartición de cambios en objetos de datos utilizando vectores de versión por objeto

La FIG. 3 ilustra un método 300 para compartir un cambio de datos en un objeto de datos en un sistema de bases de datos multimaestro usando vectores de versión por objeto, según una realización de la invención. Como se muestra, el método 300 comienza en el paso 305 donde un sitio realiza un cambio en una copia local de un objeto de datos almacenado en la copia del cuerpo de datos del sitio. Por ejemplo, un usuario puede usar una aplicación de base de datos en el sitio para agregar, eliminar o editar una o más propiedades del objeto de datos.

En una realización, como parte del cambio de un objeto de datos en un sitio, cada cambio da como resultado una nueva versión del objeto de datos en el sitio. En el paso 310, el reloj lógico local del sitio en el vector de versión para el objeto de datos se incrementa en un valor fijo (por ejemplo, uno) para reflejar la nueva versión del objeto de datos en el sitio donde se realizó el cambio. Los otros elementos en el vector de versión no se incrementan.

En una realización, cada cambio a un objeto de datos en un sitio se comparte con cualquier otro sitio en el sistema. Dependiendo de la topología del sistema multimaestro (por ejemplo, de malla completa o de malla parcial), un sitio que realiza un cambio puede comunicarse con cualquier otro sitio para compartir el cambio, o solo con un subconjunto de ellos que son responsables de comunicar el cambio con otros sitios. En el paso 315, el cambio realizado en el paso 305 se comparte con al menos otro sitio en el sistema. Compartir el cambio incluye enviar, al menos a otro sitio, datos que representan el cambio y datos que representan el vector de versión para el objeto de datos modificado después del incremento en el paso 310.

En una realización, los datos que representan el cambio incluyen un identificador del objeto de datos y una representación materializada del objeto de datos que incluye todas las propiedades del objeto de datos. En otra realización, los datos que representan el cambio incluyen un identificador del objeto de datos pero solo las propiedades del objeto de datos afectadas por el cambio. Los datos que representan el vector de versión para el objeto de datos modificado no necesitan incluir una representación de cada elemento del vector de versión y, en una realización práctica, los datos que representan solo un subconjunto de todos los elementos posibles del vector de versión se comparten con al menos otro sitio.

Detección y resolución de conflictos que involucran cambios en objetos de datos utilizando vectores de versión por objeto

La FIG. 4 ilustra un método 400 para detectar y resolver un conflicto que involucra cambios concurrentes en un objeto de datos usando vectores de versión por objeto, según una realización de la invención. Como se muestra, el método 400 comienza en el paso 405 donde un sitio recibe una actualización para un objeto de datos desde otro sitio. La actualización incluye datos que representan un cambio en el objeto de datos, incluido un identificador del objeto de datos modificado y datos que representan el vector de versión del objeto de datos modificado. El vector de versión representa la versión del objeto de datos en el sitio que realizó el cambio inmediatamente después de que se realizó el cambio. Para mayor claridad de la explicación, el vector de versión del objeto de datos modificado recibido en la actualización se denominará el vector de versión del sitio de cambio del objeto de datos.

En el paso 410, el sitio que recibe la actualización obtiene localmente su vector de versión para el objeto de datos basado en el identificador del objeto de datos incluido en la actualización y compara su vector de versión con el vector de versión del sitio de cambio para determinar si las dos versiones son idénticas, están ordenadas o son concurrentes. Como se mencionó anteriormente, esta comparación incluye comparar el vector de versión del sitio de cambio con el vector de versión del sitio receptor elemento a elemento.

En el paso 415, se determina si la versión del sitio de cambio del objeto de datos recibido en la actualización y el vector de versión del sitio receptor del objeto de datos son concurrentes. Si las dos versiones son concurrentes, se ha detectado un conflicto. Es decir, la versión del objeto de datos en el sitio receptor refleja un cambio en el objeto de datos realizado sin conocimiento del cambio recibido en la actualización y la versión del objeto de datos recibido en la actualización refleja un cambio en el objeto de datos realizado sin conocimiento del cambio del que el sitio receptor tiene conocimiento. Si se detecta un conflicto, entonces el método 400 procede al paso 420 donde los cambios concurrentes que resultan en el conflicto se resuelven bien automática o bien manualmente.

En el paso 420, se realiza una determinación inicial si el conflicto puede resolverse automáticamente. En una realización, determinar si un conflicto se puede resolver automáticamente se basa en un conjunto de heurística y/o de reglas de resolución de conflictos. El conjunto de heurística y/o de reglas de resolución de conflictos puede ser definido por el usuario. Por ejemplo, en una realización, determinar si un conflicto se puede resolver automáticamente incluye determinar si los cambios concurrentes involucran cambios en propiedades no superpuestas o en tipos de propiedades no superpuestas del objeto de datos. Por ejemplo, si el cambio recibido en la actualización es en una propiedad de Número de teléfono de un objeto de datos en particular de Persona y el cambio del que tiene conocimiento el sitio receptor es en una propiedad de Dirección del objeto de datos en particular, entonces el sistema puede determinar automáticamente que ambos cambios se pueden aceptar. En una realización, las propiedades no superpuestas se detectan en el sitio receptor al realizar una comparación propiedad a propiedad entre la versión del sitio de cambio del objeto de datos recibido en la actualización y la versión del objeto de datos del sitio receptor.

Si el conflicto no se puede resolver automáticamente, entonces el sitio receptor mantiene la actualización en una cola de actualizaciones pendientes del objeto de datos hasta que se pueda resolver el conflicto con la ayuda de la entrada del usuario. Por ejemplo, el sitio receptor puede no ser capaz de resolver automáticamente un conflicto si los cambios concurrentes involucran cambios en la misma propiedad de un objeto de datos. Por ejemplo, si el cambio recibido en la actualización es en una propiedad de Número de teléfono de un objeto de datos en particular de Persona y el cambio en el objeto de datos del que el sitio receptor tiene conocimiento es también en la propiedad de Número de teléfono de un objeto de datos en particular, entonces es posible que el sitio receptor pueda no ser capaz de resolver el conflicto automáticamente. Si bien una actualización de un objeto de datos permanece en la cola de actualizaciones pendientes del sitio receptor del objeto de datos, el sitio receptor puede continuar realizando cambios en el objeto de datos y aceptar y aplicar actualizaciones al objeto de datos recibidas de otros sitios hasta que el usuario descarte o acepte la actualización.

En una realización, para ayudar a un usuario a tomar una decisión de resolución de conflictos informada al resolver manualmente un conflicto que involucra cambios concurrentes en un objeto de datos, el sitio de resolución de conflictos determina el predecesor común más cercano en el sitio de resolución de conflictos de (a) la versión de los datos objeto en la cola de actualizaciones pendientes en el sitio de resolución de conflictos (versión pendiente) y (b) la versión actual del objeto de datos en el sitio de resolución de conflictos (versión actual). El predecesor común más cercano de estas dos versiones se determina como la versión más reciente del objeto de datos en el sitio de resolución de conflictos que está ordenado antes (es decir, sucedió antes) tanto de (a) la versión pendiente del objeto de datos como de (b) la actual versión del objeto de datos de acuerdo a sus respectivos vectores de versión. Una aplicación en el sitio de resolución de conflictos utiliza la información del predecesor común más cercano para presentar al usuario las diferencias entre ambas: (1) la versión del predecesor común más cercano del objeto de datos y la versión pendiente y (2) la versión de predecesor común más cercano y la versión actual. Por ejemplo, la aplicación puede presentar una interfaz gráfica de usuario que proporcione una indicación visual de las diferencias en cuanto a las propiedades de modo que un usuario pueda comprender la naturaleza de los cambios simultáneos e indicar qué versión del objeto de datos es correcta. En función de la presentación de las diferencias (1) y (2), el usuario puede determinar cuál de las dos versiones del objeto de datos es la versión correcta del objeto de datos y proporcionar una indicación a través de la aplicación de la versión seleccionada.

En el paso 425, la resolución de conflictos de los cambios concurrentes en el paso 420 da como resultado un cambio en la copia local del objeto de datos del sitio receptor. El cambio en el objeto de datos refleja el resultado de la resolución de conflictos. Por ejemplo, si se determinó en el paso 420 que los cambios simultáneos involucraron propiedades no superpuestas, entonces el cambio realizado en el objeto de datos en el paso 425 podría involucrar la modificación de la copia local del objeto de datos del sitio receptor para incorporar las propiedades no superpuestas modificadas recibidas en la actualización.

Después de que se realiza el cambio en la copia local del objeto de datos del sitio receptor, en el paso 430, el vector de versión del sitio de cambio del objeto de datos se unifica con el vector de versión del sitio receptor del objeto de datos. La unificación de los dos vectores de versión incluye la unificación de cada elemento en el vector de versión del sitio de cambio del objeto de datos con el elemento correspondiente en el vector de versión del sitio receptor del objeto de datos. Unificar dos elementos incluye elegir el numéricamente mayor de los dos elementos como el valor del elemento en el nuevo vector de versión. Lo que se produce mediante esta unificación en el paso 430 es un nuevo vector de versión que se ordena después del vector de versión del sitio receptor del objeto de datos y el vector de versión del sitio de cambio del objeto de datos. Dicho de otro modo, el vector de versión del sitio receptor del objeto de datos y el vector de versión del sitio de cambio ahora ocurrieron ambos antes del nuevo vector de versión. Después de que los dos vectores de versión se unifican, el vector de versión del sitio receptor del objeto de datos se reemplaza con el nuevo vector de versión que luego se convierte en el vector de versión del objeto de datos en el sitio receptor.

El paso 435 es similar a una combinación de los pasos 310 y 315 del método 300. En el paso 435, el reloj lógico del sitio receptor en el vector de versión del objeto de datos se incrementa en un valor fijo (por ejemplo, uno) para reflejar el cambio realizado en el paso 425 como resultado de la resolución de conflictos en el paso 420. Los otros elementos en el vector de versión no se incrementan. Además, en el paso 430, los cambios en la copia del objeto de datos del sitio receptor se comparten con otros sitios en el sistema.

Si, en el paso 415, el sitio receptor determina que el vector de versión del sitio de cambio del objeto de datos y el vector de versión del sitio receptor del objeto de datos son idénticos o están ordenados (es decir, no son concurrentes), entonces, en el paso 440, el sitio receptor bien incorpora la actualización en la copia local del objeto de datos del sitio receptor o bien descarta la actualización. En una realización, el sitio receptor incorpora la actualización en la copia local del objeto de datos del sitio receptor si el vector de versión del sitio receptor del objeto de datos está ordenado antes (es decir, sucedió antes) del vector de versión del sitio de cambio del objeto de datos. La incorporación de la actualización en la copia local del objeto de datos del sitio receptor incluye la sobrescritura de la información del objeto de datos en la copia local del sitio receptor con los cambios de sustitución del objeto de datos incluidos en la actualización. En una realización, el sitio receptor descarta la actualización si el vector de versión del sitio receptor del objeto de datos es idéntico al vector de versión del sitio de cambio del objeto de datos. El sitio receptor también puede descartar la actualización si el vector de versión del sitio de cambio para el objeto de

datos está ordenado antes (es decir, sucedió antes) del vector de versión del sitio receptor del objeto de datos. En este último caso, la actualización representa un cambio antiguo que ya se incorporó y fue reemplazado por la versión del sitio de recepción del objeto de datos.

Si, en el paso 435, la actualización se incorporó a la copia local del sitio de recepción del objeto de datos, entonces, en el paso 450, el vector de versión del sitio de cambio del objeto de datos se unifica con el vector de versión del sitio de recepción del objeto de datos para producir una nueva versión del vector del objeto de datos en el sitio receptor. El paso 450 es similar al paso 430. Sin embargo, a diferencia del caso en el que la actualización recibida del objeto de datos está en conflicto con la versión del sitio de recepción del objeto de datos, el nuevo vector de versión del objeto de datos en el sitio de recepción no se incrementa después de unificar el vector de versión del sitio receptor del objeto de datos y el vector de versión del sitio de cambio del objeto de datos.

El método 300 y el método 400 de las figuras 3 y 4 se explicarán ahora adicionalmente con un ejemplo con referencia a la FIG. 5. La FIG. 5 ilustra un ejemplo de intercambio y resolución de conflictos de los cambios de datos en el sistema 100 multimaestro. El tiempo lógico avanza hacia abajo desde la parte superior de la figura hasta la parte inferior a medida que tienen lugar los acontecimientos en los sitios 101, 102 y 103. Como se muestra, cada sitio 101, 102, y 103 inicialmente tiene copias idénticas del mismo objeto de datos. El objeto de datos tiene dos atributos: un atributo Tipo y un atributo Nombre. El atributo Tipo se establece en el valor "Persona" y el atributo Nombre se establece en el valor "J. S." en cada copia del objeto de datos en cada sitio. Además, cada sitio 101, 102 y 103 guarda un vector de versión del objeto de datos. Inicialmente, los vectores de versión son idénticos (es decir, <1, 0, 0>) lo que refleja que cada sitio tiene la misma versión del objeto de datos. Cada vector de versión tiene tres elementos, uno para cada sitio 101, 102 y 103. En el ejemplo representado en la FIG. 5, el primer elemento (más a la izquierda) de cada vector de versión corresponde al sitio 101, el segundo elemento (en el medio) de cada vector de versión corresponde al sitio 102, y el tercer elemento (más a la derecha) de cada vector de versión corresponde al sitio 103.

En el acontecimiento 503 en el sitio 101, se realiza un cambio local en la copia del objeto de datos del sitio 101. En particular, se cambia la propiedad Nombre de "J. S." a "John Smith". De acuerdo con el paso 310 del método 300, el reloj lógico del sitio 101 del objeto de datos se incrementa en un valor fijo. En el ejemplo, el reloj lógico del sitio 101 en el vector de versión del objeto de datos se incrementa de 1 a 2.

De acuerdo con el paso 315 del método 300, en el acontecimiento 505, el sitio 101 comparte el cambio a su copia del objeto de datos con el sitio 102. En particular, se envía una actualización desde el sitio 101 al sitio 102. En una realización, la actualización incluye un identificador del objeto de datos, datos que representan el cambio realizado y datos que representan el vector de versión del sitio 101 del objeto de datos (por ejemplo, <2, 0, 0>).

En el acontecimiento 507, la actualización enviada desde el sitio 101 se recibe en el sitio 102. De acuerdo con el paso 410 del método 400, el vector de versión para el objeto de datos recibido en la actualización <2, 0, 0> se compara con el vector de versión actual del sitio 102 del objeto de datos <1, 0, 0>. Dicha comparación revela que el vector de versión del sitio 102 ocurrió antes (está ordenado antes) del vector de versión del sitio 101. Por lo tanto, la actualización recibida en el sitio 102 que refleja el cambio realizado en el sitio 101 no entra en conflicto con la versión del objeto de datos del sitio 102. De acuerdo con el paso 440 del método 400, el sitio 102 incorpora el cambio recibido en la actualización en su copia local del objeto de datos con el cambio recibido en la actualización reemplazando cualquier propiedad diferente de la copia del objeto de datos del sitio 102. En particular, el valor de la propiedad Nombre en la copia del objeto de datos del sitio 102 se cambia de "J. S." a "John Smith". De acuerdo con el paso 450 del método 400, el vector de versión del sitio 101 del objeto de datos recibido en la actualización se unifica con el vector de versión del sitio 102 para producir un nuevo vector de versión del objeto de datos en el sitio 102 de <2, 0, 0>.

En el acontecimiento 509, la actualización del sitio 101 es propagada por el sitio 102 al sitio 103. En una realización, el sitio 102 está configurado para realizar tal propagación como parte de una topología de replicación multimaestro en cascada o de malla parcial. En una realización alternativa, en lugar de confiar en el sitio 102 para propagar la actualización, el sitio 101 comunica la actualización tanto al sitio 102 como al sitio 103 como parte de una topología de replicación multimaestro de malla completa. En el acontecimiento 511, el sitio 103 recibe la actualización e incorpora la actualización en su copia local del objeto de datos y combina vectores de versión realizando pasos similares a aquellos realizados por el sitio 102 en el acontecimiento 507.

El acontecimiento 513 y el acontecimiento 515 representan cambios concurrentes en el objeto de datos. En particular, en el sitio 102 se agrega una propiedad Número de teléfono al objeto de datos. En el sitio 103, se agrega una propiedad Dirección al objeto de datos. De acuerdo con el paso 310 del método 300, el sitio 102 y el sitio 103 incrementan los dos su reloj lógico del objeto de datos. En el acontecimiento 517, el sitio 102 envía una actualización al sitio 103 que refleja la adición de la propiedad Número de teléfono. En el acontecimiento 519, el sitio 103 envía una actualización al sitio 102 que refleja la adición de la propiedad Dirección. Aunque no se muestra en la Figura 5, los sitios 102 y 103 también pueden comunicar actualizaciones a otros sitios en el sistema (por ejemplo, al sitio 101). En el acontecimiento 521, el sitio 102 recibe la actualización enviada desde el sitio 103 y detecta el conflicto. En particular, el vector de versión recibido en la actualización del sitio 103 (es decir, <2, 0, 1>) no es idéntico ni está

ordenado antes o después del vector de versión del objeto en el sitio 102 (es decir, <2, 1, 0>). De acuerdo con el paso 420 del método 400, el sitio 102 intenta resolver el conflicto automáticamente en función de un conjunto preestablecido de heurística y/o de reglas de resolución de conflictos. En el ejemplo de la figura 5, el sitio 102 compara su copia del objeto de datos con la versión del objeto de datos recibida en la actualización y determina que los cambios concurrentes involucran cambios en propiedades no superpuestas. Por lo tanto, en el acontecimiento 512, el sitio 102 determina que el conflicto se puede resolver automáticamente y actualiza su copia local del objeto de datos en consecuencia. En particular, la propiedad Dirección recibida en la actualización se agrega a la copia local del sitio 102 del objeto de datos. Además, de acuerdo con el paso 430 del método 400, el vector de versión del sitio 102 del objeto de datos se unifica con el vector de versión del sitio 103 del objeto de datos recibido en la actualización y el vector de versión resultante se convierte en el nuevo vector de versión del objeto de datos en el sitio 102. Luego, de acuerdo con el paso 435 del método 400, el sitio 102 incrementa su reloj lógico en el vector de versión del objeto de datos en uno para producir un nuevo vector de versión del objeto de datos en el sitio 102 de <2, 2, 1>.

En el acontecimiento 523, el sitio 103 realiza un proceso similar al que realiza el sitio 102 en el acontecimiento 521.

Evitación de actualizaciones repetitivas innecesarias

Después del acontecimiento 521 en el sitio 102 y después del acontecimiento 523 en el sitio 103, el sitio 102 y el sitio 103 tienen las dos copias idénticas del objeto de datos. Sin embargo, el sitio 102 y el sitio 103 tienen diferentes vectores de versión del objeto de datos. En el ejemplo, el sitio 102 tiene un vector de versión del objeto de datos de <2, 2, 1> y el sitio 103 tiene un vector de versión del objeto de datos de <2, 1, 2>. De acuerdo con el paso 435 del método 400, el sitio 102 y el sitio 103 pueden enviarse una actualización entre sí reflejando sus respectivas operaciones de resolución de conflictos automáticas realizadas en los acontecimientos 521 y 523 respectivamente. Cuando las reciba el otro sitio, estas actualizaciones se detectarán como un conflicto. Por ejemplo, el vector de versión del sitio 102 <2, 2,1> no es idéntico, ni está ordenado antes o después que el vector de versión del sitio 103 <2, 1, 2>. Si no se toman medidas correctivas, el sitio 102 y el sitio 103 repetida e innecesariamente resolverán el conflicto, incrementarán sus relojes lógicos del objeto de datos y se enviarán actualizaciones entre sí a pesar de que ambos sitios tienen copias idénticas del objeto de datos.

En una realización, para evitar actualizaciones repetitivas innecesarias, en el paso 420 del método 400, después de que se haya detectado un conflicto, se realiza una comparación entre la versión del objeto de datos recibido en la actualización y la versión del objeto de datos del sitio receptor. Si las dos versiones son idénticas, entonces solo se realiza una unificación de los dos vectores de versión (paso 430). La copia local del sitio de recepción del objeto de datos no se cambia y el reloj lógico del sitio de recepción en el vector de versión del objeto de datos no se incrementa (es decir, no se realizan los pasos 425 y 435). En una realización, esta comparación incluye una comparación propiedad a propiedad entre las dos versiones del objeto de datos.

Por ejemplo, volviendo a la FIG. 5, en el acontecimiento 529, el sitio 102 recibe una actualización del sitio 103 que indica que el sitio 103 agregó la propiedad de Número de teléfono a su copia del objeto de datos e incluye su vector de versión actual del objeto de datos de <2, 1, 2>. Al recibir esta actualización, el sitio 102 detecta un conflicto porque su vector de versión <2, 2, 1> no es idéntico ni está ordenado antes o después del vector de versión del sitio 103 <2, 1, 2>. El sitio 102 compara su versión del objeto de datos con la versión del objeto de datos recibida en la actualización del sitio 103. Al determinar que las versiones son idénticas (es decir, ambas versiones tienen las mismas propiedades con los mismos valores), el sitio 102 unifica los dos vectores de versión para producir un nuevo vector de versión del objeto de datos en el sitio 102 de <2, 2, 2>. El sitio 103 realiza un proceso similar en el acontecimiento 531 para llegar al mismo vector de versión <2, 2, 2>. Ahora que ambos vectores de versión son idénticos, ya no se puede detectar un conflicto y las actualizaciones relacionadas con la resolución de conflictos previa ya no son propagadas por los sitios.

Vectores de versión por conjunto de vínculos

En una realización, se hace una versión de los vínculos que conectan dos objetos de datos distinta e independientemente de los objetos de datos conectados por los vínculos. En particular, el conjunto de vínculos que conectan dos objetos está asociado con su propio vector de versión distinto de los vectores de versión asociados con los dos objetos. Cada sitio guarda un vector de versión para cada conjunto de vínculos. Los cambios en un conjunto de vínculos en un sitio, incluyendo la adición de un vínculo al conjunto o la eliminación de un vínculo del conjunto, hacen que el sitio incremente su reloj lógico local del conjunto de vínculos y que el sitio comparta el cambio en el conjunto de vínculos con otros sitios. Los vectores de versiones asociados con copias de un conjunto de vínculos en los sitios se pueden usar para detectar y resolver conflictos que involucren cambios concurrentes en dos copias diferentes del mismo conjunto de vínculos de una manera similar a la descrita anteriormente de cómo se pueden usar los vectores de versión por objeto para detectar y resolver conflictos que involucren cambios concurrentes en dos copias diferentes del mismo objeto de datos.

Además, los vectores de versión por conjunto de vínculos permiten a los sitios incorporar automáticamente un cambio concurrente que incluye un cambio en un conjunto de vínculos y un cambio en un objeto de datos conectado

a otro objeto de datos mediante el conjunto de vínculos. Por ejemplo, supóngase que el sitio A y el sitio B tienen la misma versión del objeto de datos X y la misma versión del objeto de datos Y. Además, supóngase que el vector de versión del sitio A del objeto de datos X es idéntico al vector de versión del sitio B del objeto de datos X y que el vector de versión del sitio A para el objeto de datos Y es idéntico al vector de versión del sitio B para el objeto de datos Y. Si se realiza un cambio local en el objeto de datos X en el sitio A (por ejemplo, al agregar una nueva propiedad), entonces el sitio A incrementa su reloj lógico local en el vector de versión del objeto de datos X y envía una actualización al sitio B. Supóngase que, antes de que el sitio B reciba la actualización con respecto al cambio en el objeto de datos X en el sitio A, se realiza un cambio local en el sitio B que vincula el objeto de datos X y el objeto de datos Y. Según una realización, esto hace que el sitio B incremente su reloj lógico local en el vector de versión del conjunto de vínculos que conecta los objetos de datos X e Y. Sin embargo, en este caso, el sitio B no incrementa su reloj lógico local ni del objeto de datos X ni del objeto de datos Y. El sitio B luego envía una actualización al sitio A que refleja el cambio en el conjunto de vínculos entre los objetos de datos X e Y. Al recibir la actualización desde el sitio B con respecto al cambio del conjunto de vínculos, el sitio A incorpora la actualización de tal manera que el objeto de datos X según ha sido modificado por el cambio en el sitio A se vincula al objeto de datos Y. Del mismo modo, al recibir la actualización desde el sitio A con respecto al cambio en el objeto de datos X, el sitio B incorpora la actualización de tal manera que el objeto de datos X según ha sido modificado por el cambio en el sitio A se vincula al objeto de datos Y. Después de que las actualizaciones se hayan compartido entre sí, tanto el sitio A como el sitio B tienen copias idénticas del objeto de datos X y del objeto de datos Y y copias idénticas de los conjuntos de vínculos de los objetos de datos X e Y conectados.

Este ejemplo se ilustra en la FIG. 6. Según se muestra, inicialmente el sitio A y el sitio B tienen la misma versión del objeto de datos X y la misma versión del objeto de datos Y. Los acontecimientos 603 y 605 representan cambios concurrentes. En particular, en el acontecimiento 603, se realiza un cambio local en el objeto de datos X en el sitio A. Por ejemplo, se realiza un cambio que involucra una propiedad del objeto de datos X. Al mismo tiempo, en el acontecimiento 605, se realiza un cambio local en el sitio B que vincula el objeto de datos X y el objeto de datos Y. Por ejemplo, si el objeto de datos X y el objeto de datos Y representan a una persona en particular, pueden vincularse a través de una relación de Amigo de. En el acontecimiento 607, el sitio A comparte su cambio en el objeto de datos X con el sitio B e incluye su vector de versión del objeto de datos X <2, 0, 0> en su actualización. En el acontecimiento 609, el sitio B comparte su cambio en el conjunto de vínculos X-Y e incluye su vector de versión del conjunto de vínculos X-Y <1, 0, 0> en su actualización. Ambos sitios reciben e incorporan actualizaciones de cada uno en sus respectivas copias de la base de datos en los acontecimientos 611 y 613. Obsérvese que en este ejemplo no se detecta ningún conflicto entre los cambios concurrentes porque se hace una versión del conjunto de vínculos que conectan los objetos de datos X e Y distinta e independientemente de los objetos de datos X e Y mismos.

Uso de vectores de versión por objeto para detectar conflictos en la resolución de objetos

Como se mencionó, algunos sistemas de bases de datos pueden admitir "resolución de objetos". La resolución de objetos involucra a un usuario o a un proceso informático automatizado que determina que dos o más objetos de datos distintos realmente representan la misma entidad del mundo real e invoca una función del sistema de bases de datos para que los objetos de datos distintos se agrupen en un único objeto de datos. Por ejemplo, supóngase que hay dos objetos de datos distintos, uno con un valor de propiedad de Nombre de "John Smith", el otro con un valor de propiedad de Nombre de "J. S.". Un usuario puede decidir que estos dos objetos de datos representan la misma persona del mundo real. En consecuencia, en un sistema de bases de datos que admite la resolución de objetos, el usuario puede invocar una función del sistema de bases de datos para que los dos objetos de datos distintos se resuelvan en un único objeto de datos que tenga un valor de propiedad de nombre de "John Smith" o de "J. S." según lo seleccionado por el usuario que resuelve los objetos.

En los sistemas de bases de datos multimaestro que emplean replicación asíncrona, sería deseable detectar como un conflicto cambios concurrentes que incluyen un cambio en la resolución del objeto. Por ejemplo, supóngase que en la base de datos A, el usuario 1 cambió la propiedad del color del cabello de un objeto de datos que representa a una persona llamada "J. S." de "castaño" a "rubio". Además, supóngase que antes de que el cambio de color del cabello realizado por el usuario 1 pueda propagarse desde la base de datos A a la base de datos B, el usuario 2 cambia la base de datos B resolviendo el objeto de datos que representa "J. S." juntándolo con otro objeto de datos que representa a una persona llamada "John Smith". Sería deseable que el sistema de bases de datos multimaestro detecte estos dos cambios concurrentes como un conflicto ya que el usuario 2 pudiera no haber decidido resolver "J. S." y "John Smith" juntándolos si el usuario 2 hubiera sabido que el color del cabello de John Smith fue cambiado por el usuario 1. De manera similar, el usuario 1 pudiera no haber decidido cambiar el color del cabello de "J. S." si el usuario 1 hubiera sabido que el usuario 2 resolvió "J. S." y "John Smith" juntándolos.

En una realización, los vectores de versión por objeto se usan para detectar como conflicto un cambio concurrente que involucra un cambio en la resolución del objeto. En particular, cuando un sitio resuelve dos o más objetos juntándolos, el sitio incrementa cada reloj lógico local en el sitio en cada vector de versión de cada objeto de datos resuelto al juntarlo. La resolución de los objetos de datos se comparte luego como una actualización con otros sitios. La actualización incluye los vectores de versión resultantes del sitio de intercambio de cada uno de los objetos de datos que se resolvieron juntándolos.

Según una realización, un sitio que recibe la actualización detecta un conflicto comparando cada vector de versión de cada objeto de datos en la actualización de la resolución del objeto con su vector de versión del objeto de datos correspondiente. Si alguno de los vectores de versión es concurrente, se detecta un conflicto. La resolución de los objetos se incorpora a la copia de la base de datos del sitio receptor solo si todos y cada uno de los vectores de versión recibidos en la actualización es idéntico o está ordenado después del vector de versión correspondiente en el sitio receptor.

Como ejemplo, supóngase que el objeto de datos X en el sitio 101 de la FIG. 1 tiene el vector de versión <1, 0, 0> y el objeto de datos Y en el sitio 101 tiene el vector de versión <1, 0, 0>. Cuando los objetos de datos X e Y se resuelven juntándolos en el sitio 101, cada reloj lógico de los objetos de datos X e Y en el sitio 101 se incrementa en un valor fijo (por ejemplo, uno) dando un vector de versión en el sitio 101 de <2, 0, 0> del objeto de datos X y un vector de versión en el sitio 101 de <2, 0, 0> del objeto de datos Y. Cuando el cambio en la resolución del objeto en el sitio 101 es compartido por el sitio 101 con otros sitios (por ejemplo, con el sitio 102 y con el sitio 103), la actualización incluye datos que indican el cambio en la resolución del objeto (es decir, que los objetos de datos X e Y se resolvieron juntándolos) y los vectores de versión del sitio 101 de los objetos de datos que se resolvieron juntándolos (por ejemplo, <2, 0, 0> del objeto de datos X y <2, 0, 0> del objeto de datos Y). Además, supóngase que se realiza un cambio concurrente con el cambio en la resolución del objeto realizado en el sitio 101 en el objeto de datos X en el sitio 102, cambiando así el vector de versión del objeto de datos X en el sitio 102 de <1, 0, 0> a <1, 1, 0>. Por ejemplo, una propiedad del objeto de datos X se modifica en el sitio 102. Al recibir la actualización enviada desde el sitio 101 con respecto al cambio en la resolución del objeto, el sitio 102 detectará estos cambios concurrentes como un conflicto. Se detectará un conflicto en el sitio 102 porque un vector de versión para al menos un objeto de datos recibido en la actualización de la resolución del objeto desde el sitio 101 es concurrente con el vector de versión para el objeto de datos en el sitio 102. En particular, el vector de versión del objeto de datos X recibido en la actualización <2, 0, 0> es concurrente con el vector de versión del objeto de datos X en el sitio 102 <1, 1, 0>. En respuesta a la detección del conflicto, el sitio 102 puede intentar resolver el conflicto automáticamente conforme a las reglas predefinidas de heurística y/o de resolución de conflictos, o puede requerir la entrada por parte de un usuario para resolver el conflicto.

El tener conocimiento de la resolución de objetos ocurre después (RAHA)

En una realización, un sitio que recibe una actualización que involucra un cambio en un objeto de datos que se ha resuelto juntándolo en el sitio de recepción con uno o más de otros objetos de datos se aplicará en el sitio de recepción únicamente si todos y cada uno de los objetos de datos resueltos al juntarlos en el sitio receptor están disponibles en la actualización. Si todos y cada uno de los objetos de datos no están disponibles en la actualización, entonces la actualización puede colocarse en la cola de actualizaciones pendientes del sitio receptor. Un proceso en el sitio receptor inspecciona periódicamente la cola de actualizaciones pendientes en busca de actualizaciones que, cuando se combinan, incluyen todos y cada uno de los objetos de datos resueltos al juntarlos en el sitio receptor. Si el proceso de inspección descubre dicha combinación, entonces las actualizaciones se pueden aplicar atómicamente en combinación en el sitio receptor.

Por ejemplo, considérense los siguientes acontecimientos que tienen lugar en el sistema 100 de la figura 1:

(1) Tanto el sitio 101 como el sitio 102 tienen copias de los objetos de datos X, Y y Z cada uno en la versión <1, 0, 0>. Además, los objetos de datos X, Y y Z se resuelven juntándolos tanto en el sitio 101 como en el sitio 102.

(2) En el sitio 101, el objeto de datos X no se resuelve de los objetos de datos Y y Z. Cada vector de versión en el sitio 101 se incrementa de tal manera que cada objeto de datos X, Y y Z tiene ahora la versión <2, 0, 0> en el sitio 101.

(3) El sitio 101 envía una actualización al sitio 102 que incluye datos que representan el objeto de datos X en la versión <2, 0, 0> y datos que representan la resolución de los objetos de datos Y y Z cada uno en la versión <2, 0, 0>.

(4) El sitio 102 recibe la actualización del sitio 101 y la coloca en su cola de actualizaciones pendientes. La actualización se coloca en la cola de actualizaciones pendientes porque ni el objeto de datos X en la versión <2, 0, 0> ni la resolución de los objetos de datos Y y Z en la versión <2, 0, 0> incluyen todos los objetos de datos en la resolución de objetos de datos X, Y y Z cada uno en la versión <1, 0, 0> en el sitio 102.

(5) Un proceso de inspección en el sitio 102 inspecciona la cola de actualizaciones pendientes en busca de actualizaciones que, cuando se combinan, incluyen todos y cada uno de los objetos de datos X, Y y Z resueltos al juntarlos en el sitio 102. El proceso de inspección encuentra las actualizaciones recibidas desde el sitio 102 en las actualizaciones pendientes de los objetos de datos X, Y y Z y las aplica a la copia del sitio 102 de los datos del cuerpo después de lo cual tanto el sitio 101 como el sitio 102 tienen el objeto de datos X en la versión <2, 0, 0> sin resolver a partir de los objetos de datos Y y Z resueltos, cada uno en la versión <2, 0, 0>.

Vectores de versión por reconocimiento global de sitio

En una realización, para ayudar a determinar qué cambios deberían compartirse con otros sitios en el sistema, cada sitio guarda un único vector de versión de reconocimiento global que el sitio comparte periódicamente con otros sitios en el sistema. El vector de versión de reconocimiento global de un sitio refleja una unificación de todos los vectores de versión de todos los cambios aplicados con éxito a la copia local del sitio del cuerpo de datos compartido. Cuando un sitio de envío comparte un cambio con un sitio receptor, se garantiza que el sistema receptor ya ha recibido con éxito todos los cambios que están ordenados antes (es decir, que sucedieron antes) del vector de versión de reconocimiento global del sitio receptor. Por lo tanto, el sitio de envío no necesita enviar esos cambios que están ordenados antes (es decir, que sucedieron antes) al sitio de recepción del vector de versión de reconocimiento global del sitio de recepción.

En una realización, los cambios en la cola de actualizaciones pendientes en un sitio se comparten con otros sitios aunque las actualizaciones estén pendientes y aún no se hayan liberado de conflictos. Esto se hace para la corrección en sistemas en los que la topología de replicación es cíclica y/o dinámica. Por ejemplo, considérese el sistema 100 de la figura 1 en el que los tres sitios 101, 102 y 103 están configurados para compartir cambios entre sí. Considérense además los siguientes acontecimientos que tienen lugar en el sistema 100:

(1) El sitio 101 envía al sitio 102 una actualización del objeto de datos A en la versión <1, 0, 0> y una actualización del objeto de datos B en la versión <1, 0, 0>.

(2) Simultáneamente con el acontecimiento (1), el sitio 102 edita el objeto A a la versión <0, 1, 0>.

(3) El sitio 102, al recibir la actualización del objeto B en la versión <1, 0, 0,> del sitio 101, aplica la actualización a su copia local del objeto B. El sitio 102, al recibir la actualización del objeto de datos A en la versión <1, 0, 0> del sitio 101, coloca la actualización en una cola de actualizaciones pendientes en el sitio 102.

(4) El sitio 102 envía al sitio 103 una actualización del objeto de datos A en la versión <0, 1, 0> y una actualización del objeto de datos B en la versión <1, 0, 0>.

(5) El sitio 103, al recibir la actualización del objeto B en la versión <1, 0, 0> del sitio 102, aplica la actualización a su copia local del objeto B. El sitio 103, al recibir la actualización del objeto de datos A en la versión <0, 1, 0> del sitio 102, aplica la actualización a su copia local del objeto A. El vector de versión de reconocimiento global del sitio 103 está en <1, 1, 0> como resultado de la unificación del vector de versión del objeto de datos A en la versión <0, 1, 0> y el vector de versión del objeto de datos B en la versión <1, 0, 0>.

En este ejemplo, si, en el acontecimiento (4), la actualización del objeto de datos A en la versión <1, 0, 0> en la cola de actualizaciones pendientes del sitio 102 no se comparte también con el sitio 103, entonces el sitio 103 puede no recibir nunca la actualización porque el vector de versión de reconocimiento global del sitio 103 indica que el sitio 103 ya ha recibido la actualización. Por lo tanto, según una realización, el sitio 102 en el acontecimiento (4) también compartirá con el sitio 103 la actualización en su cola de actualizaciones pendientes para el objeto de datos A en la versión <1, 0, 0>. Esto es así a pesar de que aún no se ha resuelto el conflicto en la actualización. En una realización, la actualización pendiente también se almacena en la cola de actualizaciones pendientes del sitio 103. En esta situación, el conflicto ahora se puede resolver en el sitio 102 o en el sitio 103.

Mecanismos de implementación: descripción general del hardware

Según una realización, las técnicas descritas en el presente documento se implementan mediante uno o más dispositivos informáticos de propósito especial. Los dispositivos informáticos de propósito especial pueden estar cableados para realizar las técnicas, o pueden incluir dispositivos electrónicos digitales tales como uno o más circuitos integrados específicos de aplicación (ASIC) o matrices de puertas programables en campo (FPGA) que se programan de manera persistente para realizar las técnicas, o pueden incluir uno o más procesadores de hardware de propósito general programados para realizar las técnicas conforme a las instrucciones del programa en firmware, en memoria, en otro almacenamiento o en una combinación. Dichos dispositivos informáticos de propósito especial también pueden combinar lógica cableada, ASIC o FPGA personalizados con programación personalizada para llevar a cabo las técnicas. Los dispositivos informáticos de propósito especial pueden ser sistemas informáticos de escritorio, sistemas informáticos portátiles, dispositivos portátiles, dispositivos de red o cualquier otro dispositivo que incorpore lógica de programa y/o cableada para implementar las técnicas.

Por ejemplo, la FIG. 7 es un diagrama de bloques que ilustra un sistema 700 informático sobre el cual se puede implementar una realización de la invención. El sistema 700 informático incluye un bus 702 u otro mecanismo de comunicación para comunicar información, y un procesador 704 de hardware acoplado con el bus 702 para procesar información. El procesador 704 de hardware puede ser, por ejemplo, un microprocesador de propósito general.

El sistema 700 informático también incluye una memoria 706 principal, tal como una memoria de acceso aleatorio (RAM) u otro dispositivo de almacenamiento dinámico, acoplado al bus 702 para almacenar información e

instrucciones a ejecutar por el procesador 704. La memoria 706 principal también puede ser usada para almacenar variables temporales u otra información intermedia durante la ejecución de instrucciones a ejecutar por el procesador 704. Dichas instrucciones, cuando se almacenan en medios de almacenamiento accesibles para el procesador 704, convierten el sistema 700 informático en una máquina de propósito especial que está personalizada para realizar las operaciones especificadas en las instrucciones.

El sistema 700 informático incluye además una memoria 708 de solo lectura (ROM) u otro dispositivo de almacenamiento estático acoplado al bus 702 para almacenar información estática e instrucciones para el procesador 704. Se provee un dispositivo de almacenamiento 710, tal como un disco magnético o un disco óptico, y se acopla al bus 702 para almacenar información e instrucciones.

El sistema 700 informático puede estar acoplado a través del bus 702 a una pantalla 712, tal como un tubo de rayos catódicos (CRT), para mostrar información a un usuario del ordenador. Un dispositivo 714 de entrada, que incluye teclas alfanuméricas y otras teclas, está acoplado al bus 702 para comunicar información y selecciones de comandos al procesador 704. Otro tipo de dispositivo de entrada de usuario es el control 716 del cursor, tal como un ratón, una bola de seguimiento o teclas de dirección del cursor para comunicar información de dirección y selecciones de comandos al procesador 704 y para controlar el movimiento del cursor en la pantalla 712. Este dispositivo de entrada normalmente tiene dos grados de libertad en dos ejes, un primer eje (por ejemplo, x) y un segundo eje (por ejemplo, y), que permite que el dispositivo especifique posiciones en un plano.

El sistema 700 informático puede implementar las técnicas descritas en el presente documento utilizando lógica cableada, uno o más ASIC o FPGA, firmware y/o lógica de programa personalizados que, en combinación con el sistema informático, hace que el sistema 700 informático sea una máquina de propósito especial. Según una realización, las técnicas en el presente documento son realizadas por el sistema 700 informático en respuesta al procesador 704 que ejecuta una o más secuencias de una o más instrucciones contenidas en la memoria 706 principal. Dichas instrucciones pueden leerse en la memoria 706 principal desde otro medio de almacenamiento, tal como un dispositivo 710 de almacenamiento. La ejecución de las secuencias de instrucciones contenidas en la memoria 706 principal hace que el procesador 704 realice los pasos del proceso descritos en el presente documento. En realizaciones alternativas, se pueden usar circuitos cableados en lugar de o en combinación con instrucciones de software.

El término "medios no transitorios", según se usa en el presente documento, se refiere a cualquier medio que almacene datos y/o instrucciones que hagan que una máquina funcione de una manera específica. Dichos medios no transitorios pueden comprender medios no volátiles y/o medios volátiles. Los medios no volátiles incluyen, por ejemplo, discos ópticos o magnéticos, tales como el dispositivo 710 de almacenamiento. Los medios volátiles incluyen memoria dinámica, tal como la memoria 706 principal. Las formas comunes de medios no transitorios incluyen, por ejemplo, un disquete, un disco flexible, un disco duro, una unidad de estado sólido, una cinta magnética o cualquier otro medio de almacenamiento de datos magnéticos, un CD-ROM, cualquier otro medio de almacenamiento óptico de datos, cualquier medio físico con patrones de agujeros, una RAM, una PROM y EPROM, una FLASH-EPROM, una NVRAM, o cualquier otro chip o cartucho de memoria.

Los medios no transitorios son distintos pero pueden usarse junto con los medios de transmisión. Los medios de transmisión participan en la transferencia de información entre medios no transitorios. Por ejemplo, los medios de transmisión incluyen cables coaxiales, cables de cobre y fibra óptica, incluyendo los cables que componen el bus 702. Los medios de transmisión también pueden tomar la forma de ondas acústicas o de luz, tales como las generadas durante las comunicaciones de datos de ondas de radio e infrarrojos.

Varias formas de medios pueden estar involucradas en llevar una o más secuencias de una o más instrucciones al procesador 704 para su ejecución. Por ejemplo, las instrucciones pueden llevarse inicialmente en un disco magnético o unidad de estado sólido de un ordenador remoto. El ordenador remoto puede cargar las instrucciones en su memoria dinámica y enviar las instrucciones a través de una línea telefónica utilizando un módem. Un módem local para el sistema informático 700 puede recibir los datos en la línea telefónica y usar un transmisor de infrarrojos para convertir los datos en una señal de infrarrojos. Un detector de infrarrojos puede recibir los datos transportados en la señal infrarroja y los circuitos apropiados pueden colocar los datos en el bus 702. El bus 702 transporta los datos a la memoria 706 principal, desde la cual el procesador 704 recupera y ejecuta las instrucciones. Las instrucciones recibidas por la memoria 706 principal pueden almacenarse opcionalmente en el dispositivo 710 de almacenamiento antes o después de la ejecución por el procesador 704.

El sistema 700 informático también incluye una interfaz 718 de comunicación acoplada al bus 702. La interfaz 718 de comunicación proporciona un acoplamiento de comunicación de datos bidireccional a un enlace 720 de red que está conectado a una red 722 local. Por ejemplo, la interfaz 718 de comunicación puede ser una tarjeta de red digital de servicios integrados (ISDN), un módem por cable, un módem por satélite o un módem para proporcionar una conexión de comunicación de datos a un tipo correspondiente de línea telefónica. Como otro ejemplo, la interfaz 718 de comunicación puede ser una tarjeta de red de área local (LAN) para proporcionar una conexión de comunicación de datos a una LAN compatible. También se pueden implementar enlaces inalámbricos. En cualquier

implementación de este tipo, la interfaz de comunicación 718 envía y recibe señales eléctricas, electromagnéticas u ópticas que transportan flujos de datos digitales que representan diversos tipos de información.

El enlace 720 de red normalmente proporciona comunicación de datos a través de una o más redes a otros dispositivos de datos. Por ejemplo, el enlace 720 de red puede proporcionar una conexión a través de la red 722 local a un ordenador 724 anfitrión o al equipo de datos operado por un proveedor 726 de servicios de Internet (ISP). El ISP 726 a su vez proporciona servicios de comunicación de datos a través de la red mundial de comunicación de paquetes de datos ahora comúnmente conocida como "Internet" 728. La red 722 local e Internet 728 utilizan señales eléctricas, electromagnéticas u ópticas que transportan flujos de datos digitales. Las señales a través de las diversas redes y las señales en el enlace 720 de red y a través de la interfaz de comunicación 718, que transportan los datos digitales hacia y desde el sistema 700 informático, son formas ejemplares de medios de transmisión.

El sistema 700 informático puede enviar mensajes y recibir datos, incluido el código del programa, a través de la(s) red(es), del enlace 720 de red y de la interfaz 718 de comunicación. En el ejemplo de Internet, un servidor 730 podría transmitir un código solicitado para un programa de aplicación a través de Internet 728, del ISP 726, de la red 722 local y de la interfaz 718 de comunicación.

El código recibido puede ser ejecutado por el procesador 704 a medida que se recibe, y/o almacenado en el dispositivo 710 de almacenamiento, o en otro almacenamiento no volátil para su posterior ejecución.

En la especificación anterior, se han descrito realizaciones de la invención con referencia a numerosos detalles específicos que pueden variar de implementación a implementación. Por lo tanto, el indicador único y exclusivo de lo que es la invención, y que los solicitantes pretenden que sea la invención, es el conjunto de reivindicaciones que surgen de esta solicitud, en la forma específica en que se publican dichas reivindicaciones, incluida cualquier corrección posterior. Cualquier definición expresamente expuesta en el presente documento de los términos contenidos en dichas reivindicaciones regirá el significado de dichos términos tal como se utilizan en las reivindicaciones. Por lo tanto, ninguna limitación, elemento, propiedad, característica, ventaja o atributo que no se mencione expresamente en una reivindicación debe limitar el alcance de dicha reivindicación de ninguna manera. La especificación y los dibujos deben, por consiguiente, considerarse en un sentido ilustrativo más que restrictivo.

REIVINDICACIONES

1. Un método informático para compartir y resolver los conflictos de los cambios de datos en un sistema de bases de datos multimaestro que comprende una pluralidad de sitios (101, 102, 103), comprendiendo el método:

en un primer sitio de la pluralidad de sitios:

hacer (305) un primer cambio en un objeto (201) de datos en particular en una base de datos en el primer sitio para producir una primera versión del objeto de datos en particular, y

compartir (315) el primer cambio con uno o más sitios de la pluralidad de sitios;

en un segundo sitio de la pluralidad de sitios:

realizar un segundo cambio en el objeto de datos en particular en una base de datos en el segundo sitio para producir una segunda versión del objeto de datos en particular;

recibir (405) una actualización que refleja el primer cambio;

en donde la actualización incluye:

una identificación del objeto de datos en particular,

datos que reflejan el primer cambio en el objeto de datos en particular, y

un primer vector de versión que representa la primera versión del objeto de datos en particular;

obtener un segundo vector de versión de una base de datos en el segundo sitio, representando el segundo vector de versión la segunda versión del objeto de datos en particular;

comparar (415) el primer vector de versión con el segundo vector de versión para determinar si la primera versión del objeto de datos en particular y la segunda versión del objeto de datos en particular son idénticas, están ordenadas o son concurrentes;

en el primer sitio de la pluralidad de sitios:

realizar un tercer cambio en la base de datos en el primer sitio a un conjunto de uno o más vínculos que conectan dos objetos de datos, involucrando el tercer cambio agregar o eliminar uno o más vínculos del conjunto de vínculos, incluyendo los dos objetos de datos el objeto de datos en particular; y

compartir el tercer cambio con uno o más sitios de la pluralidad de sitios;

en el segundo sitio de la pluralidad de sitios:

realizar un cuarto cambio en la base de datos en el segundo sitio al conjunto de vínculos, involucrando el cuarto cambio agregar o eliminar uno o más vínculos del conjunto de vínculos;

recibir una actualización que refleja el tercer cambio;

en donde la actualización incluye:

una identificación del conjunto de vínculos,

datos que reflejan el tercer cambio, y

un tercer vector de versión que representa una versión del conjunto de vínculos en el primer sitio resultante del tercer cambio, siendo el tercer vector de versión distinto de los objetos de datos conectados por los vínculos;

obtener un cuarto vector de versión de la base de datos en el segundo sitio, representando el cuarto vector de versión una versión del conjunto de vínculos en el segundo sitio resultante del cuarto cambio, siendo el cuarto vector de versión distinto de los objetos de datos conectados por los vínculos;

comparar el tercer vector de versión con el vector de la cuarta versión para determinar si la versión del conjunto de vínculos resultante del tercer cambio y la versión del conjunto de vínculos resultante del cuarto cambio son idénticas, están ordenadas o son concurrentes; y

actualizar el vector de la cuarta versión para incorporar el tercer cambio.

2. El método según la reivindicación 1, que comprende además:

en el segundo sitio:

en respuesta a la determinación de que la primera versión del objeto de datos en particular y la segunda versión del objeto de datos en particular son concurrentes, determinando si el primer cambio y el segundo cambio tienen diferentes propiedades del objeto de datos en particular; y

en respuesta a la determinación de que el primer cambio y el segundo cambio están en diferentes propiedades del objeto de datos en particular:

incorporar el primer cambio en la segunda versión del objeto de datos en particular en la base de datos en el segundo sitio para producir una tercera versión del objeto de datos en particular, unificando el primer vector de versión y el segundo vector de versión para producir un vector de versión unificado, e

incrementar un elemento en particular del vector de versión unificado para producir un vector de versión incrementado que representa la tercera versión del objeto de datos en particular, correspondiendo el elemento en particular a un reloj lógico del objeto de datos en particular en el segundo sitio.

3. El método según la reivindicación 1, que comprende además:

en el segundo sitio:

en respuesta a la determinación de que la primera versión del objeto de datos en particular y la segunda versión del objeto de datos en particular son concurrentes, colocar la actualización recibida en el segundo sitio en una cola de actualizaciones pendientes en el segundo sitio; y

recibir la entrada por parte de un usuario que selecciona como máximo una de entre (a) la primera versión del objeto de datos en particular o (b) la segunda versión del objeto de datos en particular como la versión correcta del objeto de datos en particular.

4. El método según la reivindicación 3, que comprende además:

en el segundo sitio:

en respuesta a la entrada por parte de un usuario que selecciona la primera versión del objeto de datos en particular como la versión correcta del objeto de datos en particular:

incorporar el primer cambio en la segunda versión del objeto de datos en particular en la base de datos en el segundo sitio para producir una tercera versión del objeto de datos en particular,

unificar el primer vector de versión y el segundo vector de versión para producir un vector de versión unificado, e

incrementar un elemento en particular del vector de versión unificado para producir un vector de versión incrementado que representa la tercera versión del objeto de datos en particular, correspondiendo el elemento en particular a un reloj lógico del objeto de datos en particular en el segundo sitio.

5. El método según la reivindicación 2 o la 4, que comprende además:

en el primer sitio:

recibir una actualización en el primer sitio que refleja la tercera versión del objeto de datos en particular;

en donde la actualización recibida en el primer sitio incluye datos que representan el vector de versión incrementado; y

en respuesta a la recepción de la actualización en el primer sitio, unificar el primer vector de versión y el vector de versión incrementado recibido en la actualización para producir un vector de versión que representa la tercera versión del objeto de datos en particular.

6. El método según la reivindicación 1, que comprende además:

en el primer sitio de la pluralidad de sitios:

resolver dos o más objetos de datos juntándolos para producir un cambio en la resolución del objeto de datos en la base de datos en el primer sitio; y

compartir el cambio en la resolución del objeto de datos con uno o más sitios de la pluralidad de sitios;

en el segundo sitio de la pluralidad de sitios:

recibir una actualización que refleje el cambio en la resolución del objeto de datos realizado en el primer sitio;

en donde la actualización incluye:

una identificación de cada uno de los dos o más objetos de datos,

datos que indican que los dos o más objetos de datos se resolvieron juntándolos, y

para cada uno de los dos o más objetos de datos, un vector de versión para el objeto de datos que refleja una versión del objeto de datos en el primer sitio;

comparar, para cada uno de los dos o más objetos de datos, el vector de versión del objeto de datos recibido en la actualización con un vector de versión en el segundo sitio del objeto de datos para determinar si el cambio en la resolución del objeto de datos y una versión del objeto de datos en el segundo sitio son idénticos, están ordenados o son concurrentes;

determinar, en función de la comparación, que el cambio en la resolución del objeto de datos es concurrente con una versión en el segundo sitio de al menos uno de los dos o más objetos de datos; y

en respuesta a la determinación de que el cambio en la resolución del objeto de datos es concurrente con una versión en el segundo sitio de al menos uno de los dos o más objetos de datos, determinar que el cambio en la resolución del objeto de datos entra en conflicto con una versión en el segundo sitio de al menos uno de los dos o más objetos de datos.

7. Un medio no transitorio, legible por ordenador, que almacena instrucciones ejecutables por procesador que, cuando se ejecutan, producen la ejecución del método según se ha definido en una cualquiera de las reivindicaciones 1 a 6.

8. Un sistema de bases de datos multimaestro que comprende:

uno o más dispositivos informáticos en un primer sitio de una pluralidad de sitios de replicación;

uno o más dispositivos informáticos en un segundo sitio de la pluralidad de sitios de replicación;

comprendiendo los -uno o más- dispositivos informáticos en el primer sitio:

medios para realizar un primer cambio en un objeto de datos en particular en una base de datos en el primer sitio para producir una primera versión del objeto de datos en particular, y

medios para compartir el primer cambio con uno o más sitios de la pluralidad de sitios de replicación;

comprendiendo los -uno o más- dispositivos informáticos en el segundo sitio:

medios para realizar un segundo cambio en el objeto de datos en particular en una base de datos en el segundo sitio para producir una segunda versión del objeto de datos en particular;

medios para recibir una actualización que refleje el primer cambio;

en donde la actualización incluye:

una identificación del objeto de datos en particular,

datos que reflejan el primer cambio en el objeto de datos en particular, y

un primer vector de versión que representa la primera versión del objeto de datos en particular;

medios para obtener un segundo vector de versión de una base de datos en el segundo sitio, representando el segundo vector de versión la segunda versión del objeto de datos en particular; y medios para comparar el primer

vector de versión con el segundo vector de versión para determinar si la primera versión del objeto de datos en particular y la segunda versión del objeto de datos en particular son idénticas, están ordenadas o son concurrentes;

comprendiendo los -uno o más- dispositivos informáticos en el primer sitio de la pluralidad de sitios de replicación:

medios para hacer un tercer cambio en la base de datos en el primer sitio a un conjunto de uno o más vínculos que conectan dos objetos de datos, involucrando el tercer cambio agregar o eliminar uno o más vínculos del conjunto de vínculos, incluyendo los dos objetos de datos el objeto de datos en particular; y

medios para compartir el tercer cambio con uno o más sitios de la pluralidad de sitios de replicación;

comprendiendo los -uno o más- dispositivos informáticos en el segundo sitio de la pluralidad de sitios de replicación:

medios para realizar un cuarto cambio en la base de datos en el segundo sitio en el conjunto de vínculos, involucrando el cuarto cambio agregar o eliminar uno o más vínculos del conjunto de vínculos;

medios para recibir una actualización que refleja el tercer cambio;

en donde la actualización incluye:

una identificación del conjunto de vínculos,

datos que reflejan el tercer cambio, y

un tercer vector de versión que representa una versión del conjunto de vínculos en el primer sitio resultante del tercer cambio, siendo el tercer vector de versión distinto de los objetos de datos conectados por los vínculos;

medios para obtener un cuarto vector de versión de la base de datos en el segundo sitio, representando el cuarto vector de versión una versión del conjunto de vínculos en el segundo sitio resultante del cuarto cambio, siendo el cuarto vector de versión distinto de los objetos de datos conectados por los vínculos;

medios para comparar el vector de la tercera versión con el vector de la cuarta versión para determinar si la versión del conjunto de vínculos resultante del tercer cambio y la versión del conjunto de vínculos resultante del cuarto cambio son idénticas, están ordenadas o son concurrentes; y

medios para actualizar el vector de la cuarta versión para incorporar el tercer cambio.

9. El sistema según la reivindicación 8, comprendiendo además los -uno o más- dispositivos informáticos en el segundo sitio:

medios para determinar si el primer cambio y el segundo cambio están en diferentes propiedades del objeto de datos en particular.

10. El sistema según la reivindicación 9, comprendiendo además los -uno o más- dispositivos informáticos en el segundo sitio:

medios para incorporar el primer cambio en la segunda versión del objeto de datos en particular en la base de datos en el segundo sitio para producir una tercera versión del objeto de datos en particular;

medios para unificar el primer vector de versión y el segundo vector de versión para producir un vector de versión unificado; y

medios para incrementar un elemento en particular del vector de versión unificado para producir un vector de versión incrementado que representa la tercera versión del objeto de datos en particular, correspondiendo el elemento en particular a un reloj lógico del objeto de datos en particular en el segundo sitio.

11. El sistema según la reivindicación 10, comprendiendo además los -uno o más- dispositivos informáticos en el primer sitio:

medios para recibir una actualización en el primer sitio que refleja la tercera versión del objeto de datos en particular;

en donde la actualización recibida en el primer sitio incluye datos que representan el vector de versión incrementado; y

medios para unificar el primer vector de versión y el vector de versión incrementado recibido en la actualización para producir un vector de versión que representa la tercera versión del objeto de datos en particular.

12. El sistema según la reivindicación 8, comprendiendo además los -uno o más- dispositivos informáticos en el segundo sitio:

medios para colocar la actualización recibida en el segundo sitio en una cola de actualizaciones pendientes en el segundo sitio; y

medios para recibir la entrada por parte de un usuario que selecciona como máximo una de entre (a) la primera versión del objeto de datos en particular o (b) la segunda versión del objeto de datos en particular como la versión correcta del objeto de datos en particular.

13. El sistema según la reivindicación 12, comprendiendo además los -uno o más- dispositivos informáticos en el segundo sitio:

medios para incorporar el primer cambio en la segunda versión del objeto de datos en particular en la base de datos en el segundo sitio para producir una tercera versión del objeto de datos en particular;

medios para unificar el primer vector de versión y el segundo vector de versión para producir un vector de versión unificado; y

medios para incrementar un elemento en particular del vector de versión unificado para producir un vector de versión incrementado que representa la tercera versión del objeto de datos en particular, correspondiendo el elemento en particular a un reloj lógico del objeto de datos en particular en el segundo sitio.

14. El sistema según la reivindicación 13, comprendiendo además los -uno o más- dispositivos informáticos en el primer sitio:

medios para recibir una actualización en el primer sitio que refleja la tercera versión del objeto de datos en particular;

en donde la actualización recibida en el primer sitio incluye datos que representan el vector de versión incrementado; y

medios para unificar el primer vector de versión y el vector de versión incrementado recibido en la actualización para producir un vector de versión que representa la tercera versión del objeto de datos en particular.

FIG. 1

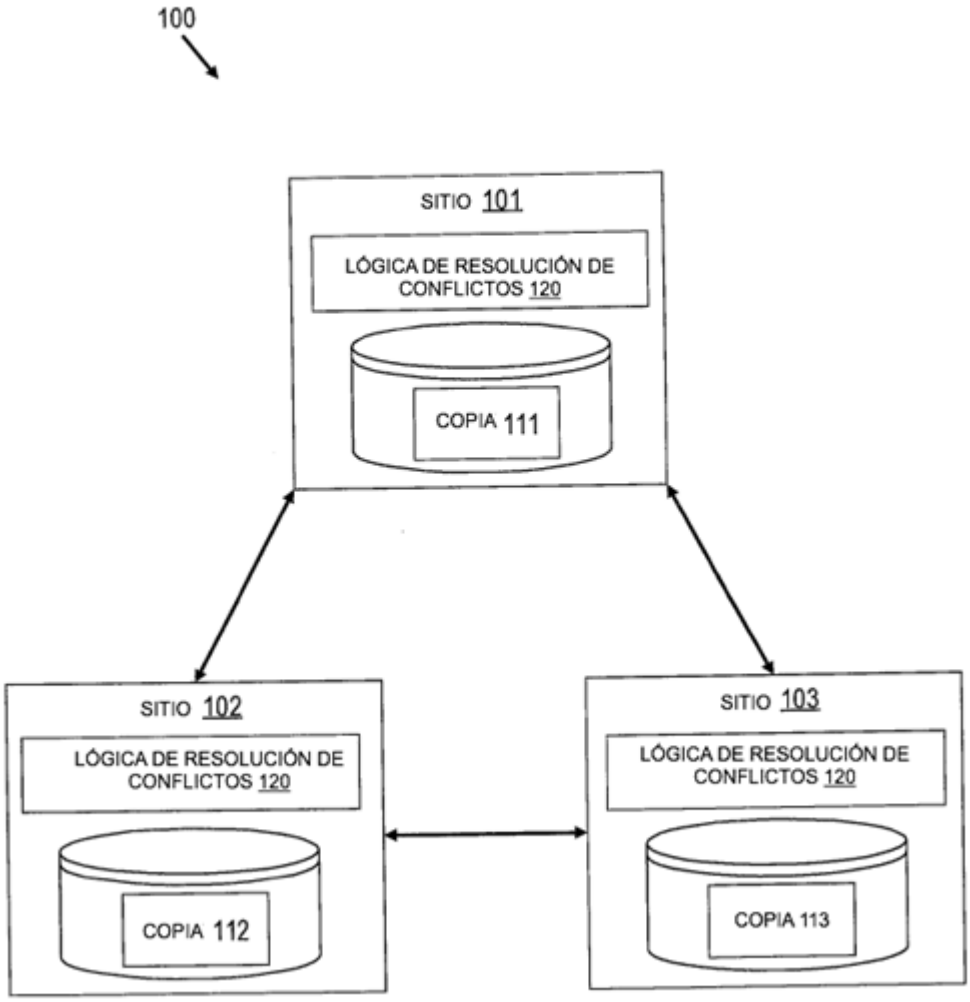

100
SITIO 101
LÓGICA DE RESOLUCIÓN DE CONFLICTOS 120
COPIA 111
SITIO 102
LÓGICA DE RESOLUCIÓN DE CONFLICTOS 120
COPIA 112
SITIO 103
LÓGICA DE RESOLUCIÓN DE CONFLICTOS 120
COPIA 113

FIG. 2

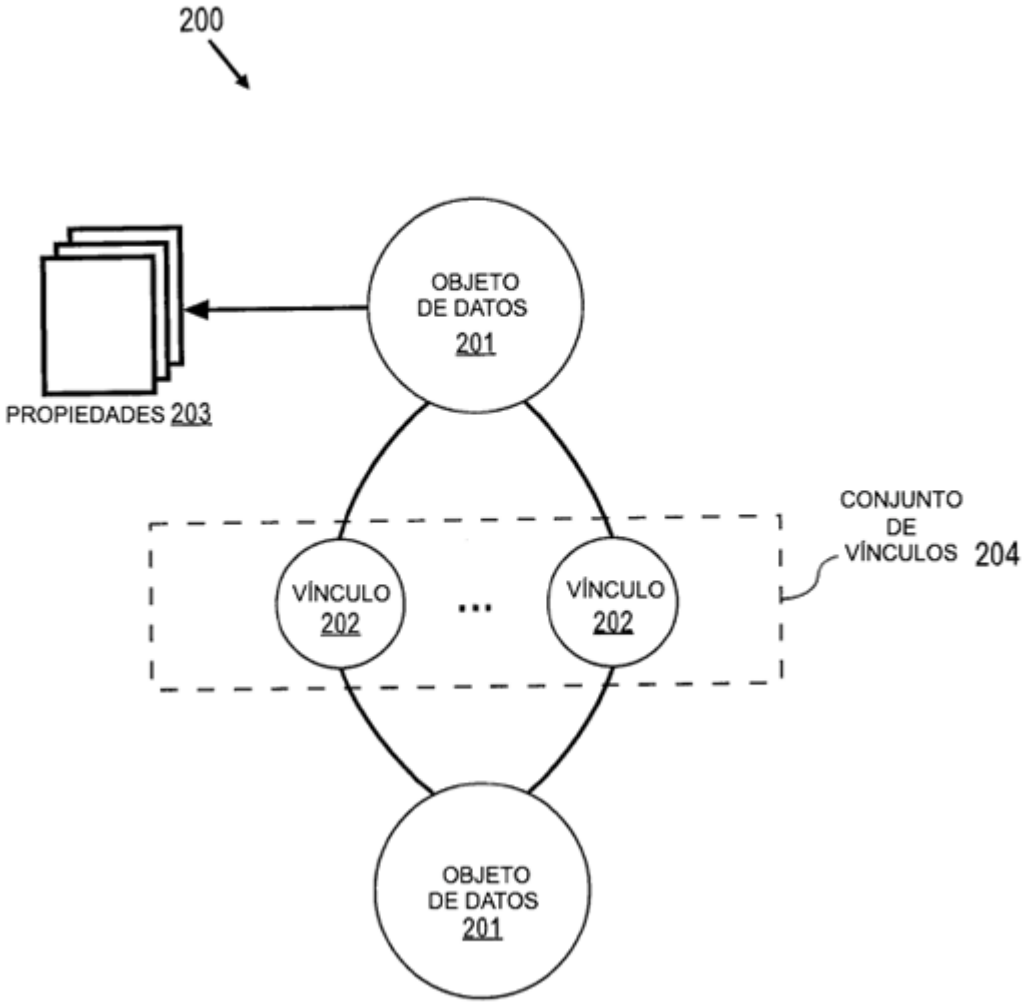

200
OBJETO DE DATOS 201
PROPIEDADES 203
CONJUNTO DE VÍNCULOS 204
VÍNCULO 202
...
VÍNCULO 202
OBJETO DE DATOS 201

FIG. 3

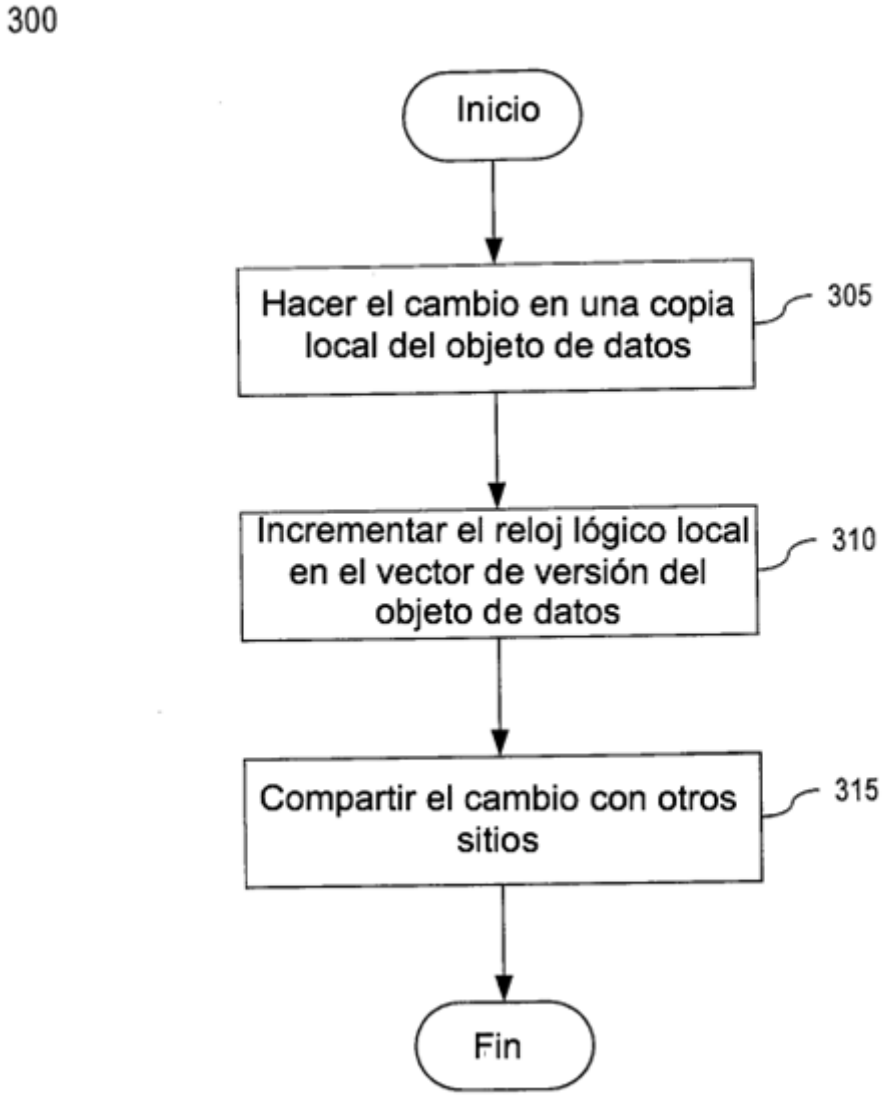

300
Inicio
Hacer el cambio en una copia local del objeto de datos
305
Incrementar el reloj lógico local en el vector de versión del objeto de datos
310
Compartir el cambio con otros sitios
315
Fin

FIG. 4

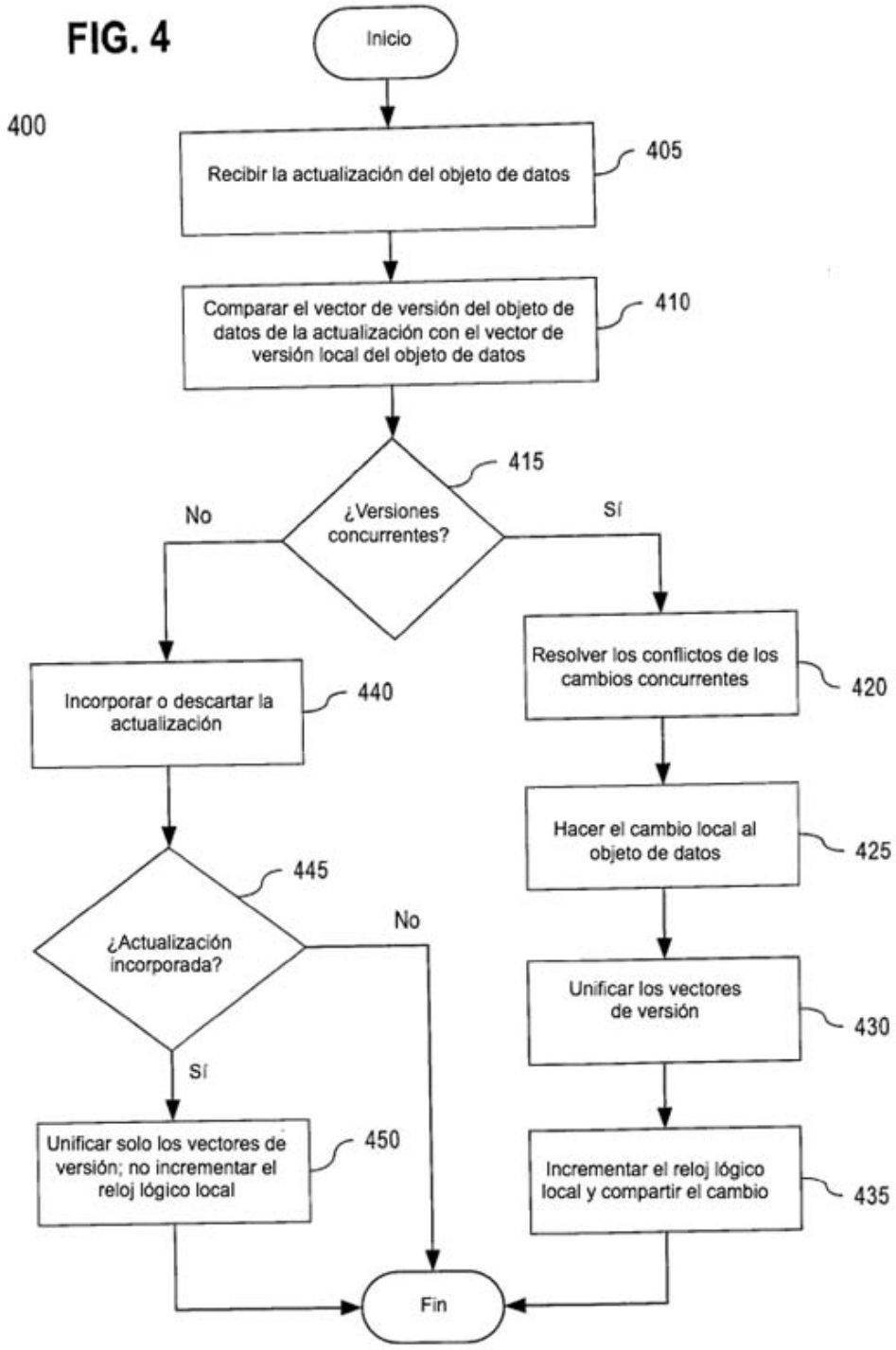

Inicio
400
Recibir la actualización del objeto de datos
405
Comparar el vector de versión del objeto de datos de la actualización con el vector de versión local del objeto de datos
410
415
¿Versiones concurrentes?
No
Sí
Incorporar o descartar la actualización
440
Resolver los conflictos de los cambios concurrentes
420
Hacer el cambio local al objeto de datos
425
445
¿Actualización incorporada?
No
Sí
Unificar los vectores de versión
430
Unificar solo los vectores de versión; no incrementar el reloj lógico local
450
Incrementar el reloj lógico local y compartir el cambio
435
Fin

FIG. 5

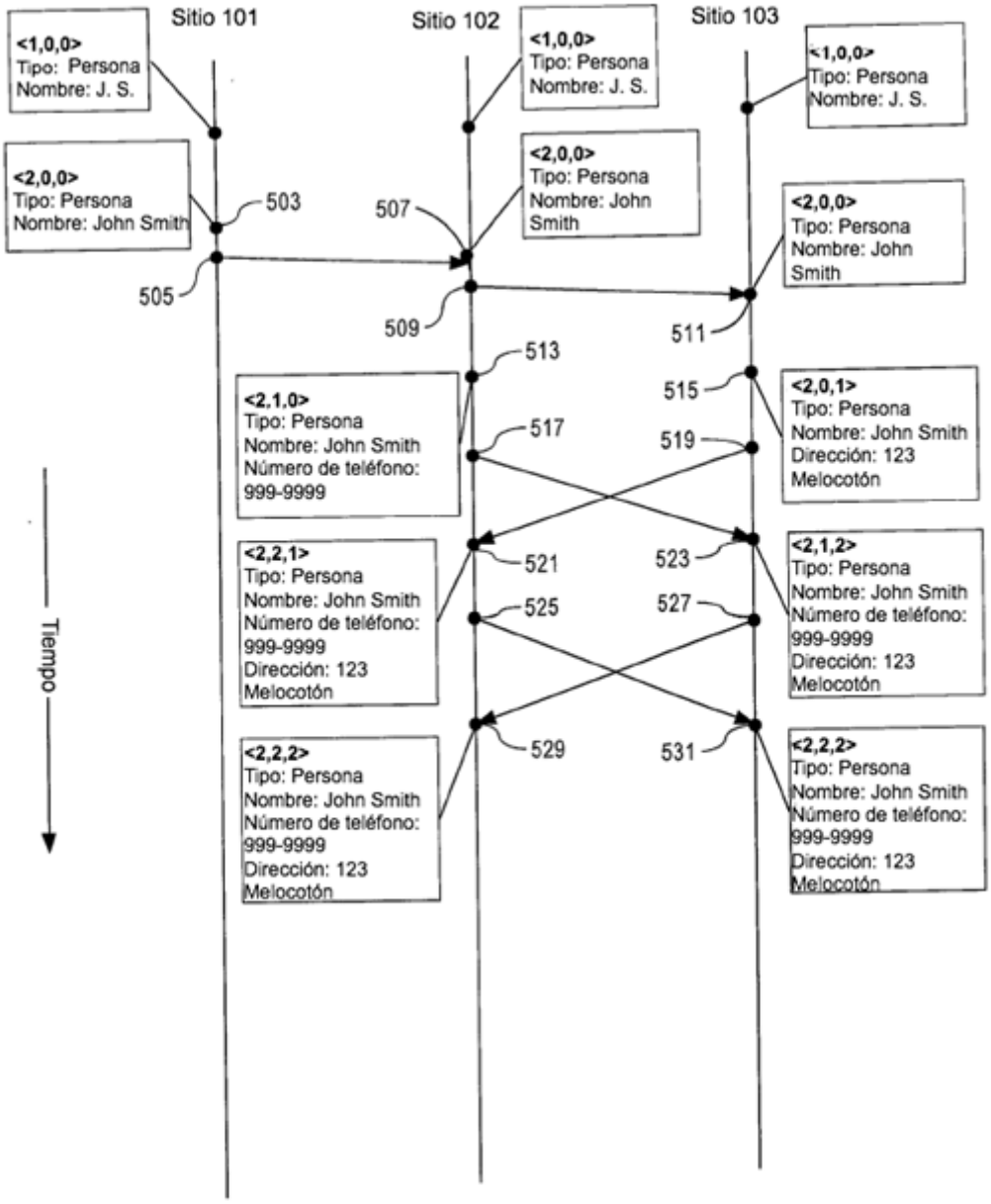

Sitio 101
Sitio 102
Sitio 103
<1,0,0>
Tipo: Persona
Nombre: J. S.
<2,0,0>
Tipo: Persona
Nombre: John Smith
503
505
507
509
511
513
515
517
519
521
523
525
527
529
531
<2,1,0>
Tipo: Persona
Nombre: John Smith
Número de teléfono:
999-9999
<2,0,1>
Tipo: Persona
Nombre: John Smith
Dirección: 123
Melocotón
<2,2,1>
Tipo: Persona
Nombre: John Smith
Número de teléfono:
999-9999
Dirección: 123
Melocotón
<2,1,2>
Tipo: Persona
Nombre: John Smith
Número de teléfono:
999-9999
Dirección: 123
Melocotón
<2,2,2>
Tipo: Persona
Nombre: John Smith
Número de teléfono:
999-9999
Dirección: 123
Melocotón
Tiempo

FIG. 6

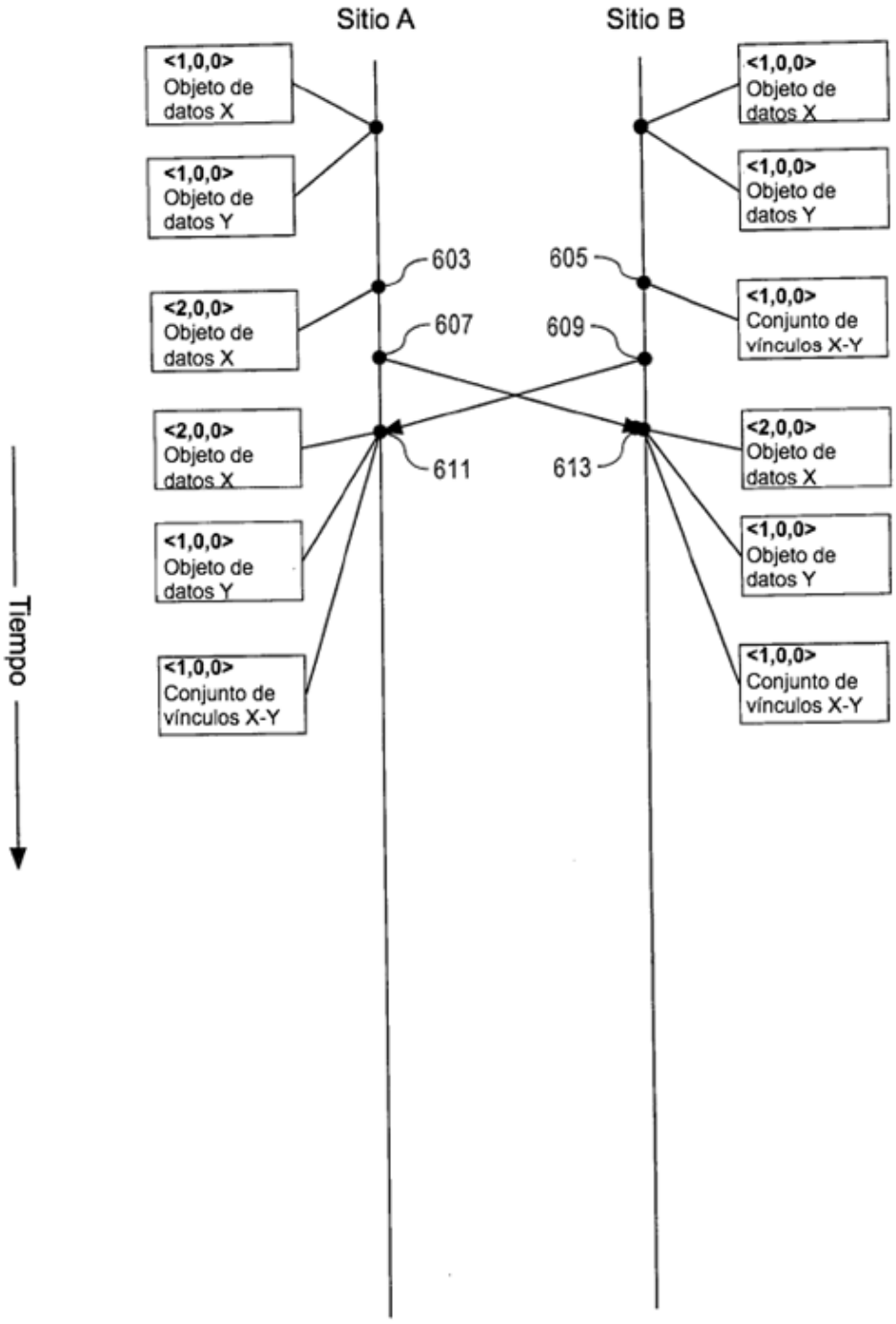
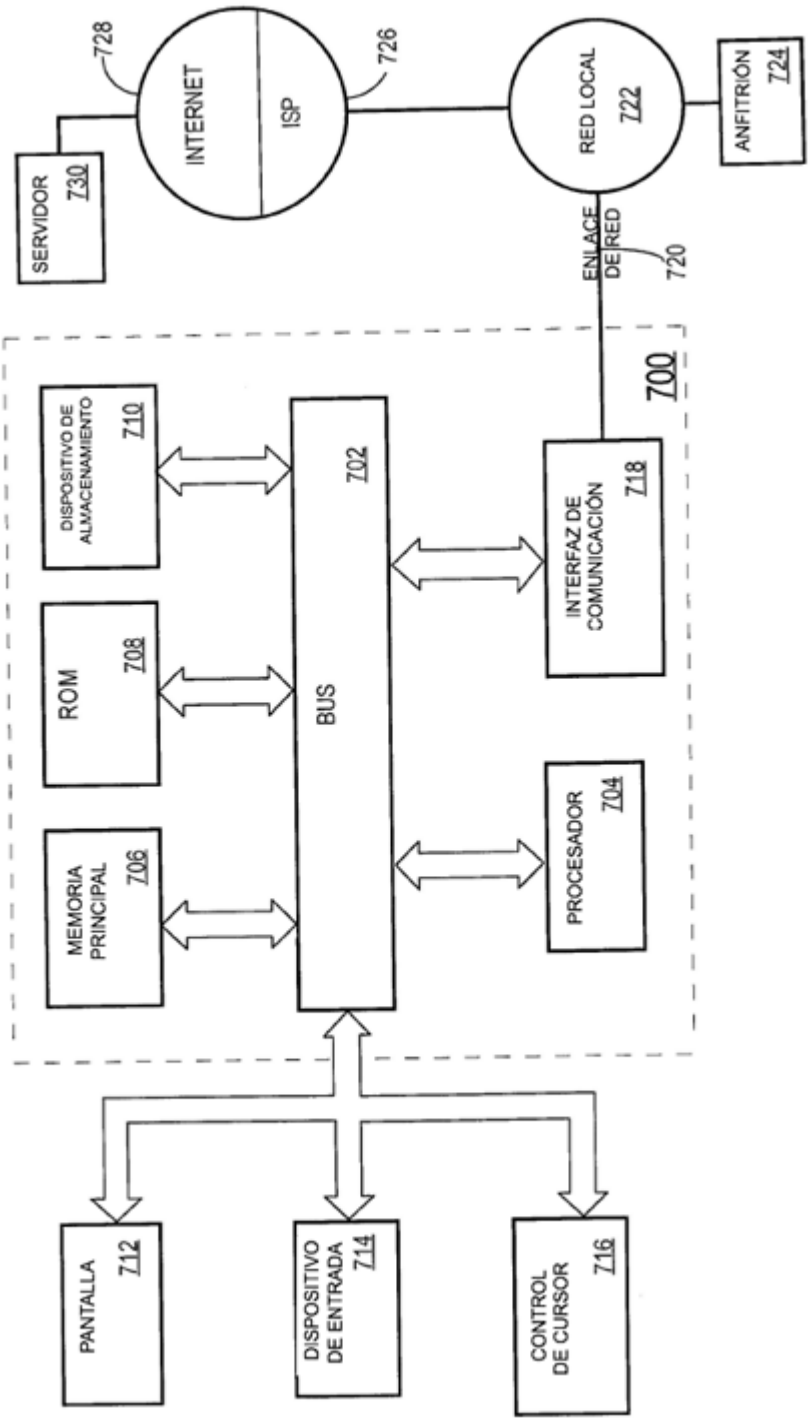

Sitio A
Sitio B
<1,0,0> Objeto de datos X
<1,0,0> Objeto de datos Y
<2,0,0> Objeto de datos X
<2,0,0> Objeto de datos X
<1,0,0> Objeto de datos Y
<1,0,0> Conjunto de vínculos X-Y
603
605
607
609
611
613
<1,0,0> Objeto de datos X
<1,0,0> Objeto de datos Y
<1,0,0> Conjunto de vínculos X-Y
<2,0,0> Objeto de datos X
<1,0,0> Objeto de datos Y
<1,0,0> Conjunto de vínculos X-Y
Tiempo

FIG. 7


728
INTERNET
ISP
726
RED LOCAL
722
ANFITRIÓN
724
SERVIDOR
730
ENLACE DE RED
720
700
DISPOSITIVO DE ALMACENAMIENTO
710
702
INTERFAZ DE COMUNICACIÓN
718
ROM
708
BUS
MEMORIA PRINCIPAL
706
PROCESADOR
704
PANTALLA
712
DISPOSITIVO DE ENTRADA
714
CONTROL DE CURSOR
716