(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 691 039**

(51) Int. CI.:

G06F 17/30 (2006.01)
G06F 12/16 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **24.03.2014 PCT/US2014/031533**

(87) Fecha y número de publicación internacional: **02.10.2014 WO14160622**

(96) Fecha de presentación y número de la solicitud europea: **24.03.2014 E 14721156 (9)**

(97) Fecha y número de publicación de la concesión europea: **25.07.2018 EP 2979203**

(54) Título: **Procesamiento de transacción usando detección de escrituras incompletas**

(30) Prioridad:

28.03.2013 US 201361806337 P
10.05.2013 US 201313892173

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
23.11.2018

(73) Titular/es:

MICROSOFT TECHNOLOGY LICENSING, LLC (100.0%)
One Microsoft Way
Redmond, WA 98052, US

(72) Inventor/es:

LARSON, PER-AKE;
FITZGERALD, ROBERT PATRICK;
DIACONU, CRISTIAN y
ZWILLING, MICHAEL JAMES

(74) Agente/Representante:

CARPINTERO LÓPEZ, Mario

DESCRIPCIÓN

Procesamiento de transacción usando detección de escrituras incompletas

Antecedentes

Los sistemas informáticos modernos incluyen uno o más procesadores que están acoplados con una memoria de sistema a través de un bus de memoria. La memoria de sistema incluye unas ubicaciones de memoria a las que se puede dirigir el procesador a través del bus de memoria. El procesador lee datos a partir de y escribe datos en la memoria de sistema por medio del bus de memoria. Por lo general, el procesador podría incluir una o más memorias caché para unas lecturas y escrituras más rápidas de los datos que se encuentran disponibles en la memoria caché.

Debido a que puede que la memoria de sistema no sea lo bastante grande para contener la totalidad de los datos e instrucciones que son necesarios, se han desarrollado algoritmos de paginación para paginar datos e instrucciones entre un almacenamiento no volátil externo (tal como una unidad de disco duro o un disco de estado sólido) y la memoria de sistema.

Los sistemas de base de datos a menudo gestionan tablas de base de datos que son bastante grandes y, por lo tanto, a menudo la estructura de tales tablas se hace persistir en el almacenamiento externo no volátil, mientras que los datos actuales sobre los que se está operando se paginan en la memoria de sistema. No obstante, algunos sistemas de base de datos más recientes almacenan tablas de base de datos en una memoria de sistema volátil. La durabilidad de tales tablas en memoria se asegura mediante el registro de todos los cambios al almacenamiento externo tal como unidades de disco duro magnéticas o unidades de disco de estado sólido. Además, tales sistemas de base de datos también pueden mantener unos puntos de comprobación del estado de la base de datos en tal almacenamiento externo. Después de un bloqueo, el último estado de la base de datos se reconstruye en memoria de sistema a partir de los últimos puntos de comprobación y el registro.

El documento US 2011/0153566 A1 se refiere a un aislamiento de instantáneas serializable optimista. El aislamiento de instantáneas se pone en práctica al llevar a cabo un seguimiento de la duración de una transacción por medio de, por ejemplo, dos marcas de tiempo: una marca de tiempo de comienzo que se asigna cuando la transacción comienza y una marca de tiempo de fin que se asigna cuando la transacción termina. Como un sistema de control de múltiples versiones, los registros no se actualizan en su sitio, más bien, la actualización o actualizaciones a un registro crean una versión nueva del registro. Una versión tiene un intervalo de tiempo válido, o tiempo de vida. En una puesta en práctica no limitante, a cada versión se le asignan dos marcas de tiempo que especifican su tiempo de vida: una marca de tiempo de comienzo de versión y una marca de tiempo de fin de versión. Una fase de validación se añade para actualizar las transacciones. Durante las operaciones normales, una transacción registra sus lecturas y sus exploraciones. Durante la fase de validación, la transacción lleva a cabo dos etapas de validación: a) el sistema vuelve a visitar las versiones que ha leído la transacción y verifica que esas versiones siguen siendo válidas a partir del fin de la transacción y b) el sistema realiza una comprobación en busca de registros fantasma mediante la repetición de la exploración de la transacción y la verificación de que no ha aparecido versión nueva alguna en la vista de la transacción desde el comienzo de la transacción. Si la transacción pasa ambas pruebas, la misma es serializable y se le permite confirmarse. Las transacciones de solo lectura no requieren una validación.

El documento de Per-Ake Larson y col., "*High-Performance Concurrency Control Mechanisms for Main-Memory Databases*", VLDB, (31 - 12 - 2011) se refiere a mecanismos de control de concurrencia de alto rendimiento para las bases de datos de memoria principal. Un sistema de base de datos optimizado para un almacenamiento en memoria puede soportar unas tasas de transacción mucho más altas que la de los sistemas actuales. Se presentan dos procedimientos de control de concurrencia eficientes específicamente diseñados para las bases de datos de memoria principal. Ambos usan un control de múltiples versiones para aislar las transacciones de solo lectura con respecto a las actualizaciones pero difieren en cuanto a cómo se logra la atomicidad: uno es optimista y uno es pesimista. Para evitar una conmutación de contexto costosa, las transacciones nunca se bloquean durante el procesamiento normal pero puede que las mismas tengan que esperar antes de la confirmación para asegurar una ordenación de serialización correcta. Además, se pone en práctica una versión optimizada para memoria principal de bloqueo de única versión. Los resultados experimentales muestran que, a pesar de que el bloqueo de única versión funciona bien cuando las transacciones son cortas y la contención es baja, el rendimiento se deteriora en unas condiciones más exigentes. Los esquemas de múltiples versiones tienen una sobrecarga más alta pero son mucho menos sensibles a las zonas activas y a la presencia de transacciones de ejecución prolongada.

El documento de Vijayan Prabhakaran y col., "*IRON file systems*", *PROCEEDINGS OF THE TWENTIETH ACM SYMPOSIUM ON OPERATING SYSTEMS PRINCIPLES*, SOSP 2005, Nueva York, Nueva York, Estados Unidos de América, (01 - 01 - 2005), ISBN 978-1-59-593079-8, página 206, se refiere a los sistemas de archivos IRON. Los sistemas de archivos de consumo confían en que los discos o bien funcionen o bien fallen completamente, si bien los discos modernos muestran unos modos de fallo más complejos. Se propone un nuevo modelo de fallos de fallo parcial para discos, que incorpora fallos localizados realistas tales como errores de sector latentes y corrupción de bloques. Se desarrolla y se aplica un marco de trabajo novedoso de creación de huellas digitales de directivas de fallos, para investigar cómo los sistemas de archivos de consumo reaccionan a una gama de fallos de disco más realistas. Sus directivas de fallo se clasifican en una taxonomía nueva que mide su robustez interna (IRON, *Internal*

RObustNess), que incluye técnicas tanto de detección de fallos como de recuperación. Se muestra que las directivas de fallo de los sistemas de archivos de consumo son a menudo inconsistentes, a veces presentan errores y, en general, son inadecuados en cuanto a su capacidad de recuperación con respecto a fallos de disco parciales. Por último, se diseña, se pone en práctica y se evalúa un prototipo de sistema de archivos IRON, Linux ixt3, mostrando que técnicas tales como realización de sumas de comprobación en disco, la replicación y la paridad potencian en gran medida la robustez del sistema de archivos al tiempo que se incurre en unas sobrecargas mínimas de tiempo y de espacio.

Breve sumario

El objeto de la presente invención es la mejora del procesamiento de transacciones en una base de datos de control de versiones múltiples.

El presente objeto se soluciona por medio de la materia objeto de las reivindicaciones independientes.

Algunas formas de realización preferidas se definen por medio de las reivindicaciones dependientes.

Al menos algunas formas de realización que se describen en el presente documento se refieren a llevar a cabo una transacción en el contexto de un sistema informático que tiene uno o más sistemas persistentes que están acoplados con uno o más procesadores a través de un bus. A modo de ejemplo, el sistema persistente puede servir como al menos parte de la memoria principal del sistema informático. La transacción podría poner en práctica un control de múltiples versiones en el que un registro no se actualiza en su sitio. Más bien, cada registro se representa como una secuencia de una o más versiones de registro, teniendo cada versión un intervalo válido durante el cual se considera que la versión de registro representa de forma apropiada el registro. El procesamiento de transacción usa una detección de escrituras incompletas de tal modo que los procesos de recuperación pueden usar tales protecciones para verificar que no hay escritura incompleta alguna. Por ejemplo, las escrituras incompletas se pueden usar para verificar la integridad de las versiones de registro así como las memorias intermedias de registro que se refieren a las versiones de registro.

El presente sumario no tiene por objeto identificar características clave o características esenciales de la materia objeto que se reivindica, ni el mismo tiene por objeto su uso como una ayuda a la hora de determinar el alcance de la materia objeto que se reivindica.

Breve descripción de los dibujos

Con el fin de describir la forma en la que se pueden obtener las ventajas y características que se han enunciado en lo que antecede, así como otras, se dará una descripción más particular de diversas formas de realización por referencia a los dibujos adjuntos. Entendiendo que estos dibujos muestran solo unas formas de realización a modo de muestra y que, por lo tanto, no se ha de considerar que sean limitantes del alcance de la invención, las formas de realización se describirán y se explicarán con una especificidad y un detalle adicionales a través del uso de los dibujos adjuntos, en los que:

la figura 1 ilustra de forma abstracta un sistema informático en el que se pueden emplear algunas formas de realización que se describen en el presente documento;
la figura 2 ilustra un entorno informático que incluye uno o más procesadores y uno o más sistemas persistentes que están acoplados por medio de un bus de comunicación;
la figura 3 ilustra un formato de versión de registro a modo de ejemplo para su uso en un sistema de control de múltiples versiones en el que los registros no se actualizan en su sitio, sino que se modifican mediante la adición de una versión de registro nueva;
la figura 4A ilustra una primera fase del procesamiento de transacción en el que un evento de comienzo de transacción desencadena un procesamiento normal;
la figura 4B ilustra una segunda fase del procesamiento de transacción en el que un evento de preconfirmación desencadena una fase de procesamiento de pre-confirmación que incluye un proceso de validación y un proceso de registro;
la figura 4C ilustra una tercera fase del procesamiento de transacción en el que un evento de confirmación / anulación desencadena una fase de post procesamiento, y se concluye por medio de un evento de terminación;
la figura 5 ilustra un diagrama de flujo de un procedimiento para llevar a cabo un procesamiento de pre-confirmación de una transacción sobre una base de datos en un sistema persistente;
la figura 6 ilustra un diagrama de flujo de un procedimiento para un procesamiento de post confirmación de la transacción;
la figura 7 ilustra un diagrama de flujo de un procedimiento para una primera parte de recuperación en un sistema informático que tiene un sistema persistente; y
la figura 8 ilustra un diagrama de flujo de un procedimiento para una segunda parte de recuperación en un sistema informático que tiene un sistema persistente.

Descripción detallada

Al menos algunas formas de realización que se describen en el presente documento se refieren a llevar a cabo una transacción en el contexto de un sistema informático que tiene uno o más sistemas persistentes que están acoplados con uno o más procesadores a través de un bus. A modo de ejemplo, el sistema persistente puede servir como al menos parte de la memoria principal del sistema informático. La transacción podría poner en práctica un control de múltiples versiones en el que un registro no se actualiza en su sitio. Más bien, cada registro se representa como una secuencia de una o más versiones de registro, teniendo cada versión un intervalo válido durante el cual se considera que la versión de registro representa de forma apropiada el registro. El procesamiento de transacción usa una detección de escrituras incompletas de tal modo que los procesos de recuperación pueden usar tales protecciones para verificar que no hay escritura incompleta alguna. Por ejemplo, las protecciones frente a escritura incompleta se pueden usar para verificar la integridad de las versiones de registro así como las memorias intermedias de registro que se refieren a las versiones de registro.

Un cierto análisis introductorio de un sistema informático se describirá con respecto a la figura 1. A continuación, el rendimiento a modo de ejemplo de una transacción y una recuperación con respecto a un fallo en un sistema informático que tiene un sistema persistente que sirve como al menos parte de la memoria principal se describirá con respecto a las figuras posteriores.

En la actualidad, los sistemas informáticos están adoptando, cada vez más, una amplia diversidad de formas. Los sistemas informáticos pueden, por ejemplo, ser dispositivos de mano, dispositivos, ordenadores portátiles, ordenadores de sobremesa, grandes sistemas, sistemas informáticos distribuidos, o incluso dispositivos que, convencionalmente, no se han considerado un sistema informático. En la presente descripción y en las reivindicaciones, la expresión “sistema informático” se define en términos generales como que incluye cualquier dispositivo o sistema (o una combinación de los mismos) que incluye al menos un procesador físico y tangible, y una memoria física y tangible capaz de tener en la misma unas instrucciones ejecutables por ordenador que pueden ser ejecutadas por el procesador. La memoria puede adoptar cualquier forma y puede depender de la naturaleza y la forma del sistema informático. Un sistema informático puede estar distribuido a lo largo de un entorno de red y puede incluir múltiples sistemas informáticos constituyentes.

Tal como se ilustra en la figura 1, en su configuración más básica, un sistema informático 100 incluye por lo general al menos una unidad de procesamiento 102 y una memoria 104. La memoria 104 puede ser una memoria de sistema física, que puede ser volátil, no volátil, o una cierta combinación de las dos. La expresión “memoria” también se puede usar en el presente documento para referirse a un almacenamiento masivo no volátil tal como un medio de almacenamiento físico. Si el sistema informático está distribuido, la capacidad de procesamiento, de memoria y / o de almacenamiento también pueden estar distribuidas. Tal como se usa en el presente documento, la expresión “módulo ejecutable” o “componente ejecutable” se puede referir a objetos de soporte lógico, encaminamientos o procedimientos que se pueden ejecutar en el sistema informático. Los diferentes componentes, módulos, motores y servicios que se describen en el presente documento se pueden poner en práctica como objetos o procesos que son ejecutados por el sistema informático (por ejemplo, como unos subprocesos separados).

En la descripción que sigue, se describen algunas formas de realización con referencia a unos actos que se llevan a cabo por medio de uno o más sistemas informáticos. Si tales actos se ponen en práctica en soporte lógico, uno o más procesadores del sistema informático asociado que lleva a cabo el acto dirigen el funcionamiento del sistema informático en respuesta a haber ejecutado unas instrucciones ejecutables por ordenador. Por ejemplo, tales instrucciones ejecutables por ordenador se pueden materializar en uno o más medios legibles por ordenador que forman un producto de programa informático. Un ejemplo de una operación de este tipo involucra la manipulación de datos. Las instrucciones ejecutables por ordenador (y los datos manipulados) se pueden almacenar en la memoria 104 del sistema informático 100. El sistema informático 100 también puede contener unos canales de comunicación 108 que permiten que el sistema informático 100 se comunique con otros procesadores de mensajes a lo largo de, por ejemplo, la red 110.

Algunas formas de realización que se describen en el presente documento pueden comprender o utilizar un ordenador de propósito especial o de propósito general que incluye soporte físico de ordenador, tal como, por ejemplo, uno o más procesadores y memoria de sistema, tal como se analiza con mayor detalle en lo sucesivo. Algunas formas de realización que se describen en el presente documento también incluyen medios legibles por ordenador físicos, y otros, para portar o almacenar unas instrucciones ejecutables por ordenador y / o estructuras de datos. Tales medios legibles por ordenador pueden ser cualesquiera medios disponibles a los que se puede acceder por medio de un sistema informático de propósito general o de propósito especial. Los medios legibles por ordenador que almacenan unas instrucciones ejecutables por ordenador son medios de almacenamiento físicos. Los medios legibles por ordenador que portan unas instrucciones ejecutables por ordenador son medios de transmisión. Por lo tanto, a modo de ejemplo y no de limitación, algunas formas de realización de la invención pueden comprender al menos dos tipos marcadamente diferentes de medios legibles por ordenador: medios de almacenamiento informático y medios de transmisión.

Los medios de almacenamiento informático incluyen RAM, ROM, EEPROM, CD-ROM u otro almacenamiento en disco óptico, almacenamiento en disco magnético u otros dispositivos de almacenamiento magnéticos, o cualquier otro medio tangible que se pueda usar para almacenar un medio de código de programa deseado en la forma de

instrucciones ejecutables por ordenador o estructuras de datos y a las que se puede acceder por medio de un ordenador de propósito general o de propósito especial.

Una “red” se define como uno o más enlaces de datos que habilitan el transporte de datos electrónicos entre sistemas informáticos y / o módulos y / u otros dispositivos electrónicos. Cuando se transfiere o se proporciona una información a lo largo de una red u otra conexión de comunicaciones (o bien cableada, o bien inalámbrica o bien una combinación de cableada o inalámbrica) a un ordenador, el ordenador ve de forma apropiada la conexión como un medio de transmisión. Los medios de transmisión pueden incluir una red y / o enlaces de datos que se pueden usar para portar un medio de código de programa deseado en la forma de instrucciones ejecutables por ordenador o estructuras de datos y a las que se puede acceder por medio de un ordenador de propósito general o de propósito especial. Las combinaciones de lo anterior también se deberían incluir dentro del alcance de los medios legibles por ordenador.

Además, tras alcanzar diversos componentes de sistema informático, un medio de código de programa en la forma de instrucciones ejecutables por ordenador o estructuras de datos se puede transferir de forma automática desde medios de transmisión a medios de almacenamiento informático (o viceversa). Por ejemplo, las instrucciones ejecutables por ordenador o estructuras de datos que se reciben a lo largo de una red o enlace de datos se pueden almacenar de forma temporal en RAM dentro de un módulo de interfaz de red (por ejemplo, un “NIC”) y, a continuación, transferirse con el tiempo a la RAM de sistema informático y / o a unos medios de almacenamiento informático menos volátiles en un sistema informático. Por lo tanto, se debería entender que se pueden incluir medios de almacenamiento informático en componentes de sistema informático que utilizan también (o incluso principalmente) medios de transmisión.

Las instrucciones ejecutables por ordenador comprenden, por ejemplo, unas instrucciones y datos que, cuando se ejecutan en un procesador, dan lugar a que un ordenador de propósito general, un ordenador de propósito especial o un dispositivo de procesamiento de propósito especial lleve a cabo una determinada función o grupo de funciones. Las instrucciones ejecutables por ordenador pueden ser, por ejemplo, códigos binarios, instrucciones de formato intermedio tales como lenguaje de ensamblador o incluso código fuente. A pesar de que la materia objeto se ha descrito en un lenguaje específico de características estructurales y / o actos metodológicos, se ha de entender que la materia objeto que se define en las reivindicaciones adjuntas no se limita necesariamente a las características descritas o los actos que se han descrito en lo que antecede. Más bien, las características y actos que se describen se divulgan como formas a modo de ejemplo de poner en práctica las reivindicaciones.

Los expertos en la materia apreciarán que la invención se puede poner en práctica en entornos informáticos de red con muchos tipos de configuraciones de sistema informático, incluyendo ordenadores personales, ordenadores de sobremesa, ordenadores portátiles, procesadores de mensajes, dispositivos de mano, sistemas de múltiples procesadores, electrónica de consumo programable o basada en microprocesador, PC de red, miniordenadores, ordenadores de gran sistema, teléfonos móviles, PDA, buscapersonas, encaminadores, conmutadores, y similares. La invención también se puede poner en práctica en entornos de sistema distribuidos en los que llevan a cabo tareas unos sistemas informáticos tanto locales como remotos, que están vinculados (o bien por medio de enlaces de datos cableados, o bien por medio de enlaces de datos inalámbricos o bien por medio de una combinación de enlaces de datos cableados e inalámbricos) a través de una red. En un entorno de sistema distribuido, los módulos de programa se pueden ubicar en dispositivos de almacenamiento en memoria tanto locales como remotos.

La figura 2 ilustra un entorno informático 200 que incluye uno o más procesadores 210 y uno o más sistemas persistentes 220 que están acoplados por medio de un bus de comunicación 230. Por ejemplo, el procesador o procesadores 210 incluyen al menos un procesador 210A, pero pueden incluir otros tal como se representa por medio de las elipses 210B. De forma similar, cualquier estructura y funcionalidad que se describa en el presente documento como atribuida al procesador 210A se puede encontrar presente y llevarse a cabo por medio de otros procesadores, en caso de haber alguno, en el procesador o procesadores 210. El sistema o sistemas persistentes 220 incluyen al menos un sistema persistente 220A, pero pueden incluir otros tal como se representa por medio de las elipses 220B. De forma similar, cualquier estructura y funcionalidad que se describa en el presente documento como atribuida al sistema persistente 220A se puede encontrar presente y llevarse a cabo por medio de otros sistemas persistentes, en caso de haber alguno, en el sistema o sistemas persistentes 220. Por ejemplo, si el entorno informático 200 fuera el sistema informático 100 de la figura, el procesador 210A puede ser el procesador 102 de la figura 1, y el sistema persistente 220A puede ser un ejemplo de la memoria 104 de la figura 1.

De acuerdo con los principios que se describen en el presente documento, el sistema persistente 220A incluye una base de datos 221. No es necesario que la base de datos 211 proporcione unos puntos de comprobación o registros que sean externos con respecto al sistema persistente 220A. Más bien, el propio sistema persistente 220A hace persistir la base de datos 221.

El procesador 210A incluye una unidad de lógica 211, unos registros 212 y una o más memorias caché 213. El entorno informático 200 también incluye un módulo de transacción 201, un módulo de recuperación 202 y un generador de marcas de tiempo 203. Las marcas de tiempo que se generan por medio del generador de marcas de tiempo 203 pueden expresar tiempo real, a pesar de que ello no se requiere. Por consiguiente, en la presente descripción, la expresión “marca de tiempo” se ha de interpretar en términos generales. Por ejemplo, el generador

de marcas de tiempo 203 podría producir simplemente unos valores monótonamente crecientes que no expresan un tiempo que no sea una ordenación temporal relativa a la granularidad de los eventos que dan lugar a los valores monótonamente crecientes. El generador de marcas de tiempo 203 puede, a petición, proporcionar el valor de marca de tiempo más reciente (es decir, leer la marca de tiempo actual) que se proporciona y / o producir un valor de marca de tiempo (es decir, hacer avanzar monótonamente la marca de tiempo) que es más grande que cualquier valor de marca de tiempo previamente generado.

Cuando se ha comenzado una transacción, el módulo de transacción 201 puede dar instrucciones al procesador 210A para llevar a cabo operaciones con el fin de instanciar un objeto de transacción que almacena una información acerca de la transacción. El procesador 210A lleva a cabo, a su vez, una transacción en la base de datos 221 tal como se especifica por medio del programa de transacción del usuario. De forma similar, el módulo de recuperación 202 puede dar instrucciones al procesador 220 para llevar a cabo operaciones con el fin de llevar a cabo una recuperación de la base de datos de una forma consistente desde el punto de vista de las transacciones (es decir, con las transacciones no confirmadas revertidas, y con las transacciones confirmadas completadas). Como alternativa, parte o la totalidad de la funcionalidad que se atribuye al módulo de transacción 201 y / o la recuperación 202 se puede incorporar en soporte físico, tal como tal vez directamente dentro del propio procesador 210A.

A medida que el procesador 210A lee a partir del sistema persistente 220A o a partir de la base de datos 221, el procesador 210A se dirige a la ubicación de los datos que se están leyendo a través del bus 230 y, de forma similar, lee los datos correspondientes a través del bus 230. El procesador 210A escribe en la memoria persistente 220A al dirigirse a la ubicación en la que se va a escribir y también mediante la provisión de los datos correspondientes que se van a escribir a través del elemento 230.

Hay tres fases asociadas con la escritura de datos en el sistema persistente 220 en una variante en memoria caché. La primera fase involucra una escritura en la que el procesador 210A escribe los datos en la memoria caché 213 (por ejemplo, a partir de uno de los registros 212). La segunda fase involucra transferir los datos desde la memoria caché 213 al sistema persistente 220A a través del bus 230. También se hará referencia a la segunda fase en el presente documento como “que proporciona” o “que transfiere” los datos al sistema persistente. La tercera fase involucra confirmar que los datos que se han proporcionado al sistema persistente de hecho se han hecho persistir. Esta persistencia podría no tener lugar de forma inmediata tras la provisión de los datos al sistema persistente. Por ejemplo, tal vez los datos se dejan en un controlador de memoria durante algún tiempo antes de hacerse persistir en la práctica. En una forma de realización, hay una orden (que se denominará una orden de “persistencia”) que da lugar a que se haga persistir cualquier dato que no se haya hecho persistir que se haya proporcionado a la memoria persistente, y devuelve la confirmación de esto mismo al emisor de la orden. Algunas puestas en práctica podrían combinar la primera y la segunda fases.

Una base de datos incluye uno o más registros. En una forma de realización, el sistema de base de datos es un sistema de base de datos de control de versiones múltiples en el que cada registro de base de datos se puede representar por medio de una secuencia de versiones de registro. Un sistema de base de datos de control de versiones múltiples no actualiza los registros en su sitio, sino que se basa, en su lugar, en el control de múltiples versiones. Una actualización de un registro crea una versión de registro completamente nueva cuyo intervalo de validez no se solapa con el de la versión de registro anterior para ese registro.

La figura 3 ilustra un formato de versión de registro a modo de ejemplo 300. El campo 301 representa un límite de comienzo de intervalo válido. Un ejemplo de un límite de este tipo es una marca de tiempo de comienzo (o BeginTS, en lo sucesivo en el presente documento). El campo indica el inicio del intervalo de tiempo válido de una versión de registro. En la forma de realización específica que se describe en lo sucesivo en el presente documento, el campo contiene o bien un identificador de transacción (ID) o bien una marca de tiempo. Cuando una transacción crea una versión de registro nueva, la transacción almacena su ID de transacción en este campo 301. Una vez que se ha confirmado la transacción, la transacción establece el campo a su marca de tiempo de confirmación.

El campo 302 representa un límite de fin de intervalo válido. Un ejemplo de un límite de este tipo es una marca de tiempo de fin (o EndTS). Este campo indica el fin del intervalo de tiempo válido de una versión de registro. Este contiene o bien un ID de transacción o bien una marca de tiempo. El campo se inicializa a un valor grande (que se denominará en lo sucesivo en el presente documento “infinito”) cuando se crea la versión de registro. El valor de “infinito” se interpreta como que significa, en esencia, que el intervalo válido no tiene un final. Cuando una transacción actualiza una versión de registro (por ejemplo, cuando se añade una versión de registro nueva que se crea a través de la actualización o la supresión de un registro), la transacción almacena en primer lugar su ID de transacción en este campo. Una vez que se ha confirmado la transacción, la transacción establece el campo de EndTS a su marca de tiempo de confirmación.

El campo 303 es un campo de encabezado de versión. Esto incluye una información de encabezado de versión que es requerida por el sistema de base de datos. Un ejemplo adecuado para algunos sistemas de base de datos son los campos de enlace. Una tabla puede tener uno o más índices. En una puesta en práctica, cada registro para esa tabla reserva un campo de enlace en cada registro que se usa para almacenar un puntero al siguiente registro en una cadena que es usada por el índice. Cuando se crea una versión nueva de un registro, la versión de registro nueva se inserta de forma inmediata en todos los índices de la tabla a la que pertenece el registro correspondiente.

El campo 304 es la carga útil, que contiene los datos de usuario y es inmutable. Es decir, cualquier actualización de la carga útil da lugar a que se cree una versión de registro nueva, en lugar de editar el contenido del campo de carga útil 304 de la versión de registro actual.

Procesamiento normal

Las figuras 4A a 4C ilustran de forma conjunta un procedimiento (al que se hace referencia en lo sucesivo en el presente documento como “procedimiento 400”) para llevar a cabo una transacción. En el presente caso, el procedimiento 400 incluye tres fases diferentes 400A, 400B y 400C de procesamiento (que se ilustran en las figuras 4A a 3C, de forma respectiva). Las tres fases 400A a 400C se agrupan por medio de cuatro eventos. El procedimiento 400 se puede llevar a cabo por medio de, por ejemplo, el módulo de transacción 201 de la figura 2.

En la figura 4A, un evento de comienzo de transacción 401A desencadena la fase de procesamiento normal 402A. El evento de comienzo de transacción 401A da lugar a la creación de un objeto de transacción con un nuevo identificador de transacción (ID) único. En una forma de realización a modo de ejemplo, el objeto de transacción tiene un estado que se establece para ser un estado activo. Además, la transacción lee la marca de tiempo actual a partir del generador de marcas de tiempo 203, marca de tiempo que representa el tiempo de lectura lógica de la transacción (y también se hará referencia al mismo en el presente documento como “tiempo de comienzo” de la transacción). Solo las versiones de registro cuyo intervalo de tiempo válido incluye el tiempo de lectura lógica de la transacción son visibles a la transacción. Se ignoran todas las otras versiones de registro.

Después de haber creado el objeto de transacción, la transacción lleva a cabo su procesamiento normal 402A, en el que la transacción podría llevar a cabo cero o más operaciones de lectura, cero o más operaciones de escritura y cero o más operaciones de exploración. Durante el procesamiento normal 402A, el objeto de transacción lleva a cabo un seguimiento del conjunto de lectura, el conjunto de escritura y el conjunto de exploración de la transacción. El conjunto de lectura de una transacción contiene una referencia (por ejemplo, un puntero) a todas las versiones de registro que se leen por medio de la transacción. El conjunto de escritura contiene una referencia (por ejemplo, un puntero) a todas las versiones de registro nuevas (a las que se hace referencia en lo sucesivo en el presente documento como “versiones de registro nuevas” o “versiones de registro recién creadas”) que se crean por medio de la transacción y todas las versiones de registro actualizadas (en lo sucesivo en el presente documento “versiones de registro hechas antiguas”) que se ha hecho que hayan dejado de ser actuales por medio de la transacción.

A modo de ejemplo, las versiones de registro recién creadas se pueden crear por medio de la transacción con una BeginTS suplente (el campo 301) que es el ID de transacción de la transacción de creación. Las versiones de registro hechas antiguas (es decir, una versión de registro que ha dejado de representar la versión más nueva de un registro debido a una versión de registro nueva que se crea por medio de la transacción) cambian su EndTS (el campo 302) de infinito a una marca de tiempo suplente que es el ID de transacción de la transacción.

En la figura 4B, un evento de preconfirmación 401B desencadena una fase de pre-confirmación 402B. El evento de preconfirmación 401B tiene lugar cuando la transacción da lugar a que el generador de marcas de tiempo 303 haga avanzar la marca de tiempo y la transacción avanza desde el estado activo a un estado de pre-confirmación. Si la transacción se confirma, la marca de tiempo que se obtiene como parte de este evento será su marca de tiempo de confirmación (que también se denomina en el presente documento “CommitTS”) y determinará la posición de las transacciones en la secuencia de confirmación de todas las transacciones.

La fase de pre-confirmación 402B consiste en dos actos - en concreto, la validación 403B y el registro 404B. Con el fin de realizar una validación 403B, la transacción valida su conjunto de lectura y su conjunto de exploración. La transacción comprueba si la misma vería exactamente las mismas versiones de registro si la totalidad de sus lecturas se llevaran a cabo a partir de la marca de tiempo de confirmación en comparación con el tiempo de lectura lógica (que se representa por medio de la marca de tiempo que se obtiene por medio de la transacción en el comienzo de la transacción en el evento 401A). El grado de validación que se requiere depende del nivel de aislamiento de la transacción.

Al igual que para el registro 404B, si la validación 403B falla, nada se registra. Si la validación 403B tiene éxito, la transacción guarda una información de procesamiento de post confirmación en una memoria intermedia de registro que incluye el identificador de transacción, un registro de confirmación que incluye una marca de tiempo, y otra información que se puede usar para llevar a cabo un procesamiento de post confirmación. A continuación, esta escribe la memoria intermedia de registro en el sistema persistente y, si la escritura se completa con éxito, la transacción se ha confirmado de manera irrevocable.

El procedimiento 400C de la figura 4C se inicia por medio de un evento de confirmación / anulación 401C. Si la validación 403B y el registro 404B se completan con éxito, la transacción cambia su estado del estado de pre-confirmación al estado confirmado. De lo contrario, si la validación 403B o el registro 404B no se completa con éxito, la transacción cambia su estado del estado de pre-confirmación a un estado anulado.

El evento de confirmación / anulación 401C inicia una fase de post procesamiento 402C. Si la transacción se anula, la transacción marca cada versión de registro nueva que creó la misma como basura mediante el establecimiento de su BeginTS a infinito y / o su EndTS a cero. La transacción establece la EndTS de cada versión de registro que la

misma hubiera hecho no actual de vuelta a infinito, de tal modo que la versión de registro antigua se puede considerar actual de nuevo.

Si la transacción se confirmó, caso en el cual se puede hacer referencia a la fase de post procesamiento 402C como la “fase de procesamiento de post confirmación”, la transacción explora el conjunto de escritura y finaliza la marca de tiempo de cada versión de registro que la misma ha insertado, actualizado o suprimido. En las versiones de registro nuevas que se crean por medio de la transacción, la transacción establece BeginTS (el campo 301) a CommitTS. En las versiones de registro que se hacen no actuales por medio de la adición de versiones de registro nuevas que se crean por medio de la transacción, la transacción establece EndTS (el campo 302) a CommitTS.

El evento de terminación 403C tiene lugar cuando el post procesamiento 402C se completa. En este evento, la transacción conmuta del estado confirmado o anulado a un estado terminado. No es necesario que una transacción de solo lectura valide su conjunto de lectura, por lo tanto la misma omite la fase de validación, se confirma de forma inmediata, y omite la fase de post procesamiento.

Una versión de registro ha dejado de ser necesaria y se puede retirar por medios lógicos de la base de datos tan pronto como su intervalo de tiempo válido ha dejado de solaparse con el tiempo de vida de cualquier transacción activa. En el presente contexto, una transacción se considera activa hasta que la misma ha completado la totalidad de su post procesamiento y su estado se ha cambiado a terminado.

La figura 5 ilustra un diagrama de flujo de un procedimiento 500 para llevar a cabo un procesamiento de pre-confirmación de una transacción actual en una base de datos en un sistema persistente. Por ejemplo, el procedimiento 500 se puede llevar a cabo por medio del procesador 210A en el contexto del entorno informático 200. La fase de procesamiento de pre-confirmación involucra un proceso de validación 403B y un proceso de registro 404B.

El procedimiento 500 involucra la canalización de trabajo que se lleva a cabo con la transacción actual, con el procesamiento de las dos transacciones anteriores. Por consiguiente, cuando el procedimiento 500 comienza a llevar a cabo la fase de pre-confirmación 402B de la transacción actual, sigue habiendo un cierto procesamiento no acabado con respecto a una transacción anterior. En concreto, la fase de post procesamiento 402C de la transacción anterior la transacción se ha de completar aún. Los intervalos de tiempo válido de las versiones de registro que se han creado o actualizado en la transacción anterior se han computado, pero aún no se ha confirmado que los mismos se han hecho persistir. Además, la memoria intermedia de registro para la transacción anterior la transacción se ha de preparar aún para su liberación de vuelta en el grupo de memorias intermedias de registro libres. Al igual que para la transacción antes de esa, la memoria intermedia de registro (que se ha preparado de forma apropiada para su liberación en el grupo de memorias intermedias de registro libres), se puede liberar en el grupo de memorias intermedias de registro libres. Por lo tanto, un subproceso de trabajo actual procesa una transacción particular en tres fases:

Fase I: Llevar a cabo un procesamiento de pre-confirmación de la transacción particular, comenzar un procesamiento de post confirmación de la transacción particular mediante la finalización (pero no la persistencia) de las marcas de tiempo de las versiones de registro que se crean o que se actualizan por medio de la transacción particular, y retardar el hecho de continuar hacia el evento de terminación hasta que se ha completado el procesamiento de post confirmación de la transacción particular;

Fase II: Al tiempo que se lleva a cabo un procesamiento de post confirmación de la fase I para una primera transacción posterior, continuar un procesamiento de post confirmación de la transacción particular al preparar la memoria intermedia de registro para la transacción particular para que se libere, y retardar el hecho de continuar hacia el evento de terminación hasta que se ha completado el procesamiento de post confirmación de la transacción particular; y

Fase III: Al tiempo que se lleva a cabo un procesamiento de post confirmación de la fase I para una segunda transacción posterior, y un procesamiento de post confirmación de la fase II para la primera transacción posterior, continuar el procesamiento de post confirmación de la transacción particular mediante la liberación de la memoria intermedia de registro para la transacción particular, y continuar hacia el evento de terminación debido a que el procesamiento de post confirmación de la transacción particular se encuentra ahora completo.

El procedimiento 500 proporciona unas protecciones frente a escritura incompleta de tal modo que el procedimiento 700 de la figura 7 puede determinar si se ha confirmado, o no, una transacción. Una escritura incompleta existe entre el instante en el que se ha emitido una orden de escritura, y el instante en el que la escritura se ha llevado a cabo sobre el objetivo duradero de la orden de escritura. Si un fallo de sistema debiera tener lugar después del instante en el que se emitiera la orden de escritura, pero tuviera lugar antes de que la escritura se haga persistir en el objetivo duradero, entonces la escritura incompleta continúa existiendo más allá del fallo de sistema. Una colección de unidades atómicas contiene una escritura incompleta si cualquiera de las unidades atómicas en la colección contiene una escritura incompleta.

Una memoria intermedia de registro para una transacción particular almacena una información suficiente para ser capaz de verificar, en el tiempo de recuperación, si la totalidad de los cambios de base de datos de una transacción se han hecho persistir. De forma similar, en la memoria intermedia de registro se almacena una información

suficiente para ser capaz de verificar si la propia memoria intermedia de registro se hizo persistir completamente. Una transacción se considera confirmada si no se detecta escritura incompleta alguna. Debido a que la presente forma de realización se basa en las protecciones frente a escritura incompleta durante la recuperación, unos pocos detalles con respecto a dos formas de realización de las protecciones frente a escritura incompleta se describirán a continuación antes de describir detalles adicionales con respecto al procedimiento 500. Se hará referencia a las dos formas de realización como la forma de realización de suma de comprobación, y la forma de realización de estado original.

La forma de realización de suma de comprobación involucra 1) computar una suma de comprobación a lo largo de la carga útil de cada versión de registro nueva y 2) añadir las sumas de comprobación a la memoria intermedia de registro de la transacción. A pesar de que las sumas de comprobación también se pueden computar y guardarse en la memoria intermedia de registro para las versiones de registro hechas antiguas de la transacción, esto no se requiere debido a que la transacción solo cambió sus marcas de tiempo, mientras que la suma de comprobación se computa a lo largo de la carga útil. Una vez que esta se ha completado para todas las versiones de registro nuevas, una suma de comprobación se computa a lo largo de los contenidos de la memoria intermedia de registro. A continuación, la suma de comprobación de la memoria intermedia de registro se puede guardar como parte de la memoria intermedia de registro.

Durante la recuperación, la suma de comprobación se computa de nuevo a lo largo de la memoria intermedia de registro (excepto por aquella porción en la que se guarda la suma de comprobación previamente computada) y se compara frente a la suma de comprobación que se guarda en la memoria intermedia de registro. Si estas sumas de comprobación coinciden, la memoria intermedia de registro se hizo persistir completamente durante una operación normal. Por consiguiente, los contenidos de la memoria intermedia de registro son válidos y, por lo tanto, la recuperación puede continuar basándose en los contenidos de la memoria intermedia de registro.

A continuación, para cada versión de registro nueva que se crea por medio de la transacción, una suma de comprobación se computa en la versión de registro y se compara frente a las sumas de comprobación guardadas para esa versión de registro en la memoria intermedia de registro. Si las mismas coinciden, la carga útil de la versión de registro nueva se hizo persistir de forma correcta. Si esto es cierto para todas las versiones de registro nuevas que se crean por medio de la transacción, las actualizaciones de la transacción se reflejan, todas ellas, en la base de datos. Por consiguiente, la transacción se puede considerar confirmada, y la recuperación puede proceder a llevar a cabo un procesamiento de post confirmación para la transacción para finalizar la transacción.

Si cualquiera de las coincidencias de suma de comprobación falla, la transacción no se confirma. En ese caso, ninguno de sus cambios se incluye en la base de datos recuperada. Esto se puede hacer mediante el restablecimiento de los marcadores de fin de intervalo válido de las versiones de registro hechas antiguas a infinito para volver a categorizar de ese modo estas versiones de registro como actuales, y mediante el establecimiento de la marca de tiempo de comienzo de las versiones de registro recién creadas al ID de transacción que se almacena en la memoria intermedia de registro, para hacer de ese modo que estas versiones de registro sean no válidas. Como alternativa, se puede renunciar a establecer las marcas de tiempo de comienzo de la versión recién creada al ID de transacción.

Cuando las sumas de comprobación se usan para la detección de escrituras incompletas, la memoria intermedia de registro incluye la siguiente información:

1) La suma de comprobación a lo largo del resto de la memoria intermedia de registro;
2) El ID de transacción;
3) La marca de tiempo de confirmación;
4) La lista de referencias (por ejemplo, punteros) a las versiones de registro hechas antiguas;
5) La lista de referencias (por ejemplo, punteros) para las versiones de registro recién creadas; y
6) La lista de sumas de comprobación, una suma de comprobación para cada versión de registro recién creada (o una única suma de comprobación si todas las versiones creadas son concatenan unas con otras).

Las sumas de comprobación no carecen de posibilidad de error. Hay una probabilidad pequeña de no detectar una escritura incompleta debido a que muchos valores diferentes se pueden poner en correspondencia con un único valor de suma de comprobación, pero esta probabilidad se puede hacer arbitrariamente pequeña mediante el uso de unas sumas de comprobación más grandes. El cómputo de las sumas de comprobación también consume ciclos de procesamiento.

En la forma de realización de estado original, antes de almacenar una versión de registro nueva en una ranura de registro vacía, la ranura de registro se llena con un patrón de bits de segundo plano previamente determinado y se hace persistir. Cuando la versión de registro se escribe en la ranura de registro vacía, la versión de registro se comprueba para ver si, por casualidad, el patrón de segundo plano se pone de manifiesto (es decir, si cualquiera de sus líneas de memoria caché contiene el patrón de segundo plano). A continuación, un mapa de bits se almacena en la memoria intermedia de registro para la transacción, en el que cada bit se corresponde con una unidad (en lo sucesivo en el presente documento, una “unidad atómica”) que se puede hacer persistir de forma atómica en el

sistema persistente. Ese bit se establece para cualquier unidad atómica que resulta que contiene el patrón de segundo plano y, de lo contrario, no se establece.

En el tiempo de recuperación, se comprueban las unidades atómicas de las versiones de registro nuevas de la transacción. Si una unidad atómica no contiene el patrón de segundo plano, se puede concluir que la unidad atómica se hizo persistir. Si la unidad atómica contiene el patrón de segundo plano, se comprueba el bit correspondiente del mapa de bits de las versiones de registro. Si el bit se establece, entonces el contenido de la unidad atómica es correcto. De lo contrario, la unidad atómica no se hizo persistir en la base de datos. La comprobación de las unidades atómicas se puede llevar a cabo por línea de memoria caché de tal modo que se comprueban al mismo tiempo múltiples unidades atómicas para la línea de memoria caché.

La misma técnica de estado original se puede aplicar a la detección de escrituras incompletas en la memoria intermedia de registro. Cuando se adquiere la memoria intermedia de registro, la memoria intermedia de registro se llena con el patrón de segundo plano y se hace persistir. Una porción de la memoria intermedia de registro (por ejemplo, la primera línea de memoria caché) se puede usar para almacenar un mapa de bits. Una vez que se ha llenado la memoria intermedia de registro, la memoria intermedia de registro se comprueba en busca de unidades atómicas que contienen el patrón de segundo plano. Para cualquier unidad atómica hallada que coincida con el patrón de segundo plano, el bit correspondiente se establece en el mapa de bits. Para asegurar que la línea de memoria caché que almacena el mapa de bits no contiene el patrón de segundo plano, el último bit del mapa de bits se puede establecer de forma consistente para que sea un valor diferente del del patrón de segundo plano.

Con un patrón de segundo plano bien escogido (por ejemplo, bits de valor cero y uno alternos), es probable que pocas versiones de registro tengan unas líneas de memoria caché que contienen el patrón de segundo plano. Para reducir el tamaño de las memorias intermedias de registro, los mapas de bits se pueden almacenar solo para las versiones de registro que sí contienen una línea de memoria caché con el patrón de segundo plano. En el presente caso, tal vez un bit disponible (que se denomina en lo sucesivo en el presente documento un “bit que se ha destinado a otro fin”) se puede usar a partir de la estructura de puntero que contiene el puntero para cada versión de registro nueva se puede usar para indicar si la versión de registro nueva tiene un mapa de bits asociado.

Cuando se usa la técnica de estado original para la detección de escrituras incompletas, la memoria intermedia de registro puede contener la siguiente información:

1) El mapa de bits que cubre la memoria intermedia de registro (puede ser una línea de memoria caché);
2) El ID de transacción;
3) La marca de tiempo de confirmación;
4) La lista de referencias (por ejemplo, punteros) a las versiones de registro hechas antiguas;
5) La lista de referencias (por ejemplo, punteros) para versiones nuevas, tal vez un bit que se ha destinado a otro fin para indicar la presencia de un mapa de bits asociado; y
6) Una lista de mapas de bits, como máximo uno para cada versión de registro recién creada (o un único mapa de bits para la totalidad de las versiones de registro recién creadas de forma conjunta).

Una ventaja de la forma de realización de estado original sobre la forma de realización de suma de comprobación es que se usa menos sobrecarga de procesamiento durante las operaciones normales. La comprobación de las líneas de memoria caché es más rápida que el cómputo de las sumas de comprobación. El enfoque de estado original también usa menos espacio debido a que la mayor parte de las versiones de registro nuevas no necesitarán un mapa de bits. No obstante, este tiene un inconveniente: la necesidad de prellenar las ranuras de registro con el patrón de segundo plano y hacer persistir el mismo. Por razones de facilidad de presentación, el mismo se describió en lo que antecede como si este prellenado se hiciera justo antes de llenar una ranura de registro con una versión de registro nueva. Esto supone un despilfarro y requiere un montón de persistencias separadas para el sistema persistente 210A. No obstante, esta desventaja se puede mitigar en gran medida al prellenar y hacer persistir las ranuras de registro en bruto antes de que se necesiten las mismas. Esto se puede hacer, por ejemplo, cuando las ranuras de registro se liberan para su reutilización. Esto ya se hace en (pequeños) lotes de tal modo que una operación de persistencia al final del lote se amortizará a lo largo de múltiples ranuras.

Con esta comprensión de las formas de realización de protección frente a escritura incompleta, a continuación la presente descripción vuelve a la descripción del procedimiento 500 de la figura 5. El proceso de validación 403B valida las exploraciones y el conjunto de lectura en la medida en la que lo requiera el nivel de aislamiento de la transacción. Si la validación tiene éxito, la transacción comienza su proceso de registro 404B. El registro 404B consiste en hacer persistir los cambios de la transacción a un sistema persistente (por ejemplo, el sistema de persistencia 220A), construir la memoria intermedia de registro y hacer persistir la memoria intermedia de registro también en el sistema persistente. En la figura 5, el proceso de validación 403B se representa por medio de los actos dentro de las llaves 510.

El proceso de validación 510 valida las lecturas y las exploraciones en la medida en la que lo requiera el nivel de aislamiento de la transacción. Por ejemplo, haciendo referencia a la figura 5, el procedimiento 500 valida una o más operaciones de lectura de la transacción (el acto 511). Por ejemplo, la transacción verifica que las versiones de registro que se leen en cada operación de lectura permanecen visibles a la transacción si la operación de lectura se

repitiera en el tiempo de confirmación de la transacción en comparación con el tiempo de comienzo (es decir, el tiempo de lectura lógica) de la transacción.

El procedimiento 500 también valida una o más operaciones de exploración de la transacción (el acto 512). Por ejemplo, la transacción verifica que las versiones de registro visibles a cada operación de exploración son las mismas, con independencia de si la exploración se lleva a cabo en el tiempo de confirmación de la transacción o el tiempo de comienzo (es decir, el tiempo de lectura lógica) de la transacción.

Si la validación falla (“No” en el bloque de decisión 513), la transacción se anula (el acto 514). Si la transacción se anula, la transacción marca cada versión de registro nueva que se crea por medio de la transacción de tal modo que el intervalo válido indica que la versión de registro ha dejado de ser válida. Por ejemplo, la BeginTS se podría establecer a cero y la ranura de registro devolverse a su lista libre. La transacción también marca cada versión de registro antigua que se ha hecho que haya dejado de ser actual por medio de la transacción, de tal modo que la versión de registro antigua es actual de nuevo. Por ejemplo, la transacción podría establecer la EndTS de tales versiones de registro de vuelta a infinito, de tal modo que la versión de registro antigua se puede considerar actual de nuevo.

Por otro lado, si la validación tiene éxito (“Sí” en el bloque de decisión 513), el procedimiento de procesamiento de pre-confirmación 500 entra en el proceso de registro que se representa por medio de los actos que son englobados por las llaves 520, y que se representa por medio del elemento 404B de la figura 4.

Una memoria intermedia de registro se asocia con la transacción actual (el acto 521). Se hará referencia a esta memoria intermedia de registro como una “memoria intermedia de registro actual” debido a que la misma se asocia con la transacción actual. En una puesta en práctica, se facilita un grupo de memorias intermedias de registro para ayudar con las transacciones que se encuentran pendientes de tal modo que se puede incluir una información de procesamiento de post confirmación para ayudar a concluir la transacción para que se termine de forma apropiada. Cada memoria intermedia de registro podría tener uno de tres estados; un estado libre, un estado de llenado y un estado lleno. Cualquier memoria intermedia de registro que tiene un estado libre se encuentra dentro del grupo de memorias intermedias de registro de las que se puede hacer uso. La transacción localiza una memoria intermedia de registro que tiene un estado libre, y cambia el estado de libre a llenado, haciendo de ese modo que el mismo sea la memoria intermedia de registro actual. Además, un ID de transacción que se asocia con la transacción se puede incluir dentro de la memoria intermedia de registro.

A continuación, la transacción almacena una información de procesamiento de post confirmación en la memoria intermedia de registro (el acto 522). Esto incluye un marcador de confirmación (por ejemplo, CommitTS) para la transacción, y una referencia (tal como un puntero) para cada versión de registro que se crea por medio de la transacción y para cada versión de registro que se actualiza por medio de la transacción. La información de procesamiento de post confirmación contiene una información que es necesaria para completar un procesamiento de post confirmación durante la recuperación si esto se vuelve necesario.

Además, para cada versión de registro que se recién crea por medio de la transacción, se lleva a cabo el contenido del recuadro 523. Por ejemplo, una protección frente a escritura incompleta se crea a lo largo de la carga útil (el campo 304) de la versión de registro nueva (el acto 524) y se escribe en la memoria intermedia de registro (el acto 525). Por ejemplo, en la forma de realización de suma de comprobación, una suma de comprobación de la carga útil se calcula y se proporciona en la memoria intermedia de registro actual de una forma que se asocia con la versión de registro recién creada correspondiente. En la forma de realización de estado original, se determina si resulta que cualquier línea de memoria caché de la versión de registro recién creada coincide, o no, con el patrón de segundo plano. De no ser así (lo que sería probablemente la mayor parte de los casos), entonces el bit que se ha destinado a otro fin (el bit de la estructura que incluye el puntero a la versión de registro nueva) puede reflejar que no hay mapa de bits alguno para la versión de registro nueva. Si es así, entonces el bit que se ha destinado a otro fin puede reflejar que hay un mapa de bits, y el mapa de bits se puede incluir en la memoria intermedia de registro actual, con el bit o bits apropiados (que se corresponden con la línea o líneas de memoria caché que resulta que incluye el patrón de segundo plano) siendo establecidos.

Una vez que la memoria intermedia de registro actual se ha llenado con la totalidad de la información de procesamiento de post confirmación que incluye el marcador de confirmación, y las referencias a las versiones de registro nuevas y actualizadas (el acto de referencia 522), y las protecciones frente a escritura incompleta (el acto de referencia 525), una protección frente a escritura incompleta se crea para la memoria intermedia de registro actual (el acto 526). En la forma de realización de suma de comprobación, esto se puede lograr al calcular una suma de comprobación a lo largo de la memoria intermedia de registro actual e incluir esa suma de comprobación calculada dentro de la memoria intermedia de registro actual. En la forma de realización de estado original, podría haber un mapa de bits asociado con la memoria intermedia de registro actual, en el que hay un bit que se corresponde con cada línea de memoria caché de la memoria intermedia de registro actual. Si la línea de memoria caché correspondiente resulta que coincide con el patrón de segundo plano, entonces el bit se establece y, de lo contrario, no se establece.

A continuación, la protección frente a escritura incompleta de nivel de transacción se asocia con la información de procesamiento de post confirmación (es decir, la memoria intermedia de registro actual) de la transacción actual (el acto 527A). Esto se puede lograr mediante la inclusión de la protección frente a escritura incompleta que se computa a lo largo de la memoria intermedia de registro actual en la propia memoria intermedia de registro actual. A continuación, la memoria intermedia de registro actual se marca como llena.

A continuación, la información de procesamiento de post confirmación y la protección frente a escritura incompleta asociada se proporcionan al sistema persistente 220A (el acto 527B). Esto se puede llevar a cabo mediante la provisión de la memoria intermedia de registro actual al sistema persistente 220A en el caso de la protección frente a escritura incompleta asociada de la memoria intermedia de registro que está incluida con la propia memoria intermedia de registro actual. Cuando se opera sobre datos y se proporcionan tales datos al sistema persistente, el procesamiento 210A puede operar en unidades de datos que se denominan “líneas de memoria caché”. Por lo tanto, cuando se proporciona la memoria intermedia de registro actual al sistema persistente 220A, el procesador 210A proporciona las líneas de memoria caché que engloban la memoria intermedia de registro actual al sistema persistente 220A. Cualquier dato sobre el que esté operando el procesador 210A se puede encontrar en una única línea de memoria caché, o puede englobar múltiples líneas de memoria caché.

En la presente descripción y en las reivindicaciones, cuando un elemento de datos (tal como una versión de registro, una memoria intermedia de registro, o una porción o porciones de la misma) se transfiere o se proporciona al sistema persistente 220A, esto no quiere decir necesariamente que el elemento de datos se haga persistir de forma inmediata en el sistema persistente 220A. Por ejemplo, cuando se proporciona un elemento de datos al sistema persistente, el elemento de datos se podría proporcionar a una memoria intermedia de almacenamiento de un controlador de memoria para el sistema persistente 220A. Por lo tanto, si tiene lugar un fallo de sistema entre el instante en el que el elemento de datos se proporciona al sistema persistente 220A y el instante en el que el elemento de datos se hace duradero en el sistema persistente 220A, el elemento de datos no se encontrará disponible tras la recuperación.

En el proceso de registro, para cada versión de registro que se crea por medio de la transacción, la versión de registro recién creada se proporciona al sistema persistente 220A (el acto 528). Para cada versión de registro que se actualiza por medio de la transacción (es decir, cada versión de registro que se hace no actual por medio de la transacción), una porción de la versión de registro se proporciona al sistema persistente (también el acto 528). La provisión al sistema persistente se puede llevar a cabo en cualquier momento después de que el procesador 210A haya completado el cómputo de la protección frente a escritura incompleta para la versión de registro correspondiente, e incluso antes si el procesador conserva una copia de la versión de registro.

Si esa versión de registro recién creada está distribuida a lo largo de múltiples líneas de memoria caché, la totalidad de esas líneas de memoria caché se proporcionan a partir del procesador 210A al sistema persistente 220A. Cuando se actualiza una versión de registro antigua para reflejar que la misma ha dejado de ser actual, solo es necesario que se cambie el intervalo de tiempo válido, y de hecho solo la EndTS (el campo 302) necesita un cambio. Por consiguiente, tal vez podría haber solo una porción de la versión de registro sobre la que se esté operando dentro de la memoria caché de procesador 213. Esta EndTS se podría encontrar presente incluso en una única línea de memoria caché. Por consiguiente, solo aquella línea de memoria caché que contiene la EndTS se proporcionaría al sistema persistente 220A.

A continuación, el proceso de registro confirma que los datos que se han proporcionado a la memoria persistente de hecho se han hecho persistir (el acto 529) en el sistema persistente. Tal como se ha mencionado previamente, hay una forma de realización en la que los elementos de datos que se han proporcionado al sistema persistente no se hacen persistir necesariamente de forma inmediata en el sistema persistente 220A. No obstante, la transacción puede emitir una orden que da lugar a que cualquier dato que se ha proporcionado al sistema persistente que aún no se ha hecho duradero, se haga de hecho duradero (o se haga persistir) en el sistema persistente. Por ejemplo, una orden de este tipo podría dar lugar a que se haga persistir cualquier dato dentro de una memoria intermedia de almacenamiento del sistema persistente 220A en el sistema persistente 220A. Una vez que ha tenido lugar esta persistencia, la orden vuelve, permitiendo que la transacción concluya que la totalidad de los datos se han hecho duraderos en el sistema persistente 220A. En la presente descripción, se hará referencia a esta orden como una orden de “persistencia”.

La figura 6 ilustra un diagrama de flujo de un procedimiento 600 para un procesamiento de post confirmación de la transacción actual. El procedimiento 600 se puede llevar a cabo por medio de la transacción después de, por ejemplo, llevar a cabo el procedimiento 500 de la figura 5. Durante el post procesamiento, se finalizan los cambios de intervalo de tiempo.

Para cada versión de registro que se recién crea o que se hace antigua por medio de la transacción, la transacción actualiza el intervalo válido usando la información de procesamiento de post confirmación (el acto 601). A modo de ejemplo, en el contexto de la figura 3, para cada versión de registro recién creada, la transacción actualiza el límite de comienzo de intervalo válido (por ejemplo, el campo 301) (el acto 611) usando la información de procesamiento de post confirmación para reflejar el tiempo de confirmación (CommitTS) de la transacción. En el contexto de la figura 3, para cada versión de registro antigua que se hace no actual por medio de la transacción, la transacción

actualiza un límite de fin de intervalo válido (por ejemplo, el campo 302) (el acto 612) usando la información de procesamiento de post confirmación. A continuación, el intervalo válido actualizado se proporciona al sistema persistente (el acto 613). Llegados a este punto, la transacción no confirma que los intervalos válidos actualizados se han hecho persistir en el sistema persistente.

En la orden de persistencia del acto 529, se puede confirmar que un número de elementos de datos se hace persistir en el sistema persistente 220A. Por ejemplo, en el acto 527B, la memoria intermedia de registro actual con la protección frente a escritura incompleta se proporcionó al sistema persistente. La orden de persistencia confirma que esto se hizo persistir de forma duradera en el sistema persistente. En el acto 528, las versiones de registro recién creadas y actualizadas se proporcionaron al sistema persistente y, por lo tanto, la orden de persistencia también confirma que estas versiones de registro nuevas y actualizadas se hacen duraderas en el sistema persistente.

No obstante, en la figura 5 no se muestra un número de otros elementos de datos que se proporcionaron al sistema persistente en una iteración anterior del procedimiento 500 para una transacción anterior. Esta orden de persistencia única también da como resultado la confirmación de que estos datos previamente proporcionados también se hicieron persistir de forma duradera dentro del sistema persistente 220A. Por ejemplo, a continuación se ha confirmado que las actualizaciones de intervalo válido de las versiones de registro que se recién crean o que se actualizan por medio de la transacción anterior se hacen persistir en el sistema persistente. Además, en el caso de que se la protección frente a escritura incompleta se pone en práctica usando la forma de realización de estado original, a continuación las memorias intermedias de registro que se asocian con la segunda transacción anterior se han restablecido con el patrón de segundo plano, habiéndose confirmado que su estado libre se hace persistir.

En el resto del procedimiento 600, se puede llevar a cabo la limpieza de memoria intermedia de registro 630 adicional que se asocia con las transacciones anteriores. Si hay una transacción previa y esta es la forma de realización de estado original (y, por lo tanto, una memoria intermedia de registro previa asociada) (“Sí” en el bloque de decisión 621), la memoria intermedia de registro previa se llena con el patrón de segundo plano y se establece a un estado libre (el acto 623) y se proporciona al sistema persistente 220A (el acto 624). Los actos 623 y 624 no son necesarios en la forma de realización de suma de comprobación. Esto se podría llevar a cabo debido a que la memoria intermedia de registro previa ha dejado de ser necesaria debido a que se confirmó que los intervalos de tiempo válido de las versiones de registro nuevas y actualizadas se han hecho duraderos en el acto 529. Llegados a este punto, no se ha confirmado aún que esa memoria intermedia de registro previa con el patrón de segundo plano y el estado libre se han hecho duraderos en el sistema persistente.

Asimismo, si hay una segunda transacción previa (y, por lo tanto, una segunda memoria intermedia de registro previa) (“Sí” en el bloque de decisión 622), se habría confirmado que el patrón de segundo plano y el estado libre de esta segunda memoria intermedia de registro previa se han hecho duraderos en el acto 529. Por consiguiente, la segunda memoria intermedia de registro previa es ahora parte del grupo de memorias intermedias de registro libres, y la segunda memoria intermedia de registro previa se reconocerá como libre por medio del gestor de recuperación. Por lo tanto, la segunda memoria intermedia de registro previa se puede añadir a la lista de memorias intermedias de registro libres. Además, la segunda transacción previa se puede retirar del mapa de transacción (el acto 625).

Por lo tanto, lo que se ha descrito es un procedimiento para procesar una transacción de principio a fin, cuando la transacción lleva a cabo un procesamiento sobre las versiones de registro dentro de un sistema persistente.

Procesamiento de recuperación

A pesar de que los principios que se describen en el presente documento no se limitan al mecanismo de recuperación particular, se describirá a continuación un procedimiento de recuperación a modo de ejemplo con respecto a las figuras 7 y 8. La figura 7 ilustra un diagrama de flujo de un procedimiento 700 para un post procesamiento que se lleva a cabo para cada transacción pendiente como una primera parte de recuperación en un sistema informático que tiene uno o más sistemas persistentes que están acoplados por medio de un bus de sistema con uno o más procesadores. La figura 8 ilustra un diagrama de flujo de un procedimiento 800 para una segunda parte de recuperación en el sistema informático, incluyendo la segunda parte reconstruir un índice y liberar las ranuras de registro no usadas. Por ejemplo, los procedimientos 700 y 800 se pueden llevar a cabo en el módulo de recuperación 202 en el contexto del entorno informático 200 de la figura 2.

El procedimiento 700 se lleva a cabo para cada memoria intermedia de registro que se halla que no se encuentra en un estado libre (por ejemplo, no es parte del grupo de memorias intermedias de registro libres). Recuérdese que cada memoria intermedia de registro no libre se asocia con una transacción. Esa asociación será evidente para el proceso de recuperación. Por ejemplo, el propio ID de transacción se podría incluir dentro de la memoria intermedia de registro.

Se accede al estado de la memoria intermedia de registro y la protección frente a escritura incompleta de nivel de transacción a partir de la memoria intermedia de registro (el acto 701). Por ejemplo, en la forma de realización de suma de comprobación, se accede a la suma de comprobación que se computa a lo largo del resto de la memoria intermedia de registro a partir de la memoria intermedia de registro. En la forma de realización de estado original, se

accede al mapa de bits que se computa a lo largo del resto de la memoria intermedia de registro a partir de la memoria intermedia de registro.

A continuación, la protección frente a escritura incompleta de nivel de transacción (o de nivel de memoria intermedia de registro) se usa para verificar si la memoria intermedia de registro (es decir, la información de procesamiento de post confirmación) contiene, o no, escritura incompleta alguna (el bloque de decisión 702) y que la memoria intermedia de registro se marca como “llena”. Si la memoria intermedia de registro no se marca como “llena” o si hay escrituras incompletas en la memoria intermedia de registro (“Sí” en el bloque de decisión 702), entonces el procedimiento 700 determina que esta transacción no se confirmó (el acto 703). Los cambios que se llevaron a cabo por medio de la transacción se desharán cuando se lleve a cabo el procedimiento 800 que se describe adicionalmente en lo sucesivo.

Si la memoria intermedia de registro se marca como “llena” y si no hay escritura incompleta alguna en la memoria intermedia de registro (“No” en el bloque de decisión 702), entonces se crea una suposición rebatible de que la transacción se confirmó (el acto 704). A continuación, el procedimiento 700 continúa determinando si hay un rebatimiento para esta suposición cuando se comprueba cada una de las versiones de registro que se actualizan como parte de la transacción. Por consiguiente, el procedimiento 700 lleva a cabo el contenido del recuadro 710 para cada una de las versiones de registro que se crean por medio de la transacción.

Para cada una de las versiones de registro creadas, el módulo de recuperación identifica la versión de registro correspondiente que se crea por medio de la transacción usando la información de procesamiento de post confirmación (el acto 711). A continuación, la protección frente a escritura incompleta de nivel de versión de registro asociada que se corresponde con la versión de registro nueva se identifica usando la información de procesamiento de post confirmación (el acto 712).

A continuación, la protección frente a escritura incompleta de nivel de versión de registro correspondiente se usa para confirmar si la versión de registro correspondiente tiene, o no, escritura incompleta alguna (el bloque de decisión 713). Por ejemplo, en la forma de realización de suma de comprobación, una suma de comprobación se computa a lo largo de la carga útil de la versión de registro. Esta se compara con la suma de comprobación que se asocia con la versión de registro (que se computó en el acto 524). Si las mismas coinciden, entonces no hay escritura incompleta alguna en la versión de registro. De no ser así, entonces hay escrituras incompletas en la versión de registro.

En la forma de realización de estado original, cada unidad atómica de la carga útil de la versión de registro se analiza para ver si la misma contiene el patrón previamente determinado. Si ninguna de las unidades atómicas de la carga útil de la versión de registro contiene el patrón de segundo plano, entonces no hay escritura incompleta alguna en la versión de registro. Si hay una o más unidades atómicas de la carga útil de la versión de registro que contienen el patrón de segundo plano, entonces se necesita un análisis adicional para determinar si la presencia del patrón de segundo plano quiere decir que hay una escritura incompleta, o que simplemente resulta que una de las unidades atómicas que se escriben en la versión de registro simplemente resulta que tiene el patrón de segundo plano.

Para determinar esto, el mapa de bits que se asocia con la versión de registro se puede revisar para ver si la unidad atómica correspondiente tiene un bit establecido (caso en el cual se escribió el patrón de segundo plano). Si el bit se establece, entonces no hay escritura incompleta alguna en esa unidad atómica. Si el bit no se establece, entonces hay una escritura incompleta en la unidad atómica (así como en la versión de registro). Recuérdese, no obstante, la forma de realización más específica en la que el puntero a la versión de registro incluye un bit que se ha destinado a otro fin que indica si la versión de registro tiene siquiera, o no, un mapa de bits correspondiente. Si no hay mapa de bits correspondiente alguno, entonces se supone que ninguna de las unidades atómicas de la versión de registro coincide con el patrón de segundo plano. Por consiguiente, si una unidad atómica que coincide con el patrón de segundo plano se halla en una versión de registro, y el bit que se ha destinado a otro fin indica que no hay mapa de bits alguno para la versión de registro, entonces esta unidad atómica representa una escritura incompleta.

Sobre la base del análisis de la protección frente a escritura incompleta de nivel de versión de registro, si hay una escritura incompleta (“Sí” en el bloque de decisión 713), entonces la suposición rebatible del acto 704 se ha rebatido, y el módulo de recuperación determina que la transacción no se confirmó (el acto 703). Por otro lado, si no hay escritura incompleta alguna que se halle como parte de este análisis (“No” en el bloque de decisión 713), entonces la suposición rebatible de que la transacción se ha confirmado no se ha rebatido y, por lo tanto, la determinación por defecto continúa siendo que la transacción se confirmó.

Si ninguna de las comprobaciones de las versiones de registro revela escrituras incompletas (“Sí” en el bloque de decisión 715), entonces la transacción se ha confirmado. En el presente caso, se identifica cada versión de registro recién escrita (que se creó por medio de la transacción o que se hizo antigua por medio de la transacción) (el acto 720). A continuación, el intervalo de tiempo válido se actualiza para cada versión de registro identificada usando la información de procesamiento de post confirmación (el acto 721). Por ejemplo, para cada versión de registro hecha antigua que se hace no actual por medio de la transacción, el límite de fin de intervalo válido (por ejemplo, el campo 302) se actualiza usando el marcador de confirmación (el acto 722). A modo de ejemplo, el límite de fin de intervalo válido se puede sustituir con CommitTS. Para cada versión de registro recién creada que se crea por medio de la

transacción, el límite de comienzo de intervalo válido (por ejemplo, el campo 301) se actualiza usando el marcador de confirmación (el acto 723). A modo de ejemplo, el límite de comienzo de intervalo válido se sustituye con CommitTS. A continuación, el intervalo válido actualizado para cada una de las versiones de registro escritas se proporciona al sistema persistente (el acto 724). A continuación de lo anterior, se confirma que el intervalo válido actualizado se ha hecho persistir en el sistema persistente (el acto 725), tal como por medio de una orden de persistencia. A continuación, la memoria intermedia de registro para esta transacción se puede marcar como “libre”.

Esto completa la primera fase de la recuperación, que es llevar a cabo el post procesamiento de cada una de las transacciones para actualizar de ese modo el intervalo de tiempo válido. En la fase final de la recuperación, la base de datos se recupera al último estado válido. Esto se logra al llevar a cabo tres tareas: 1) identificar todas las versiones de registro que son actuales y, por lo tanto, debería ser parte del estado de la base de datos, 2) identificar todas las porciones de registro (que también se denominan en el presente documento “ranuras” o “ranuras de registro”) que están vacías o contienen unas versiones de registro que no deberían ser parte del estado de la base de datos, y 3) reconstruir los índices de la base de datos se reconstruyen. Por lo tanto, la figura 8 ilustra un diagrama de flujo de un procedimiento 800 para recuperar la base de datos, reconstruir los índices y liberar las ranuras de registro no usadas. El procedimiento 800 se puede llevar a cabo para cada una de al menos algunas porciones que se corresponden con la base de datos 221 en el sistema persistente 220A. En una forma de realización, las porciones constituyen unas ranuras de registro. Se podría indicar que una ranura de registro está vacía, y no tiene una versión de registro, si el límite de comienzo de intervalo válido indica, por ejemplo, cero. De lo contrario, una ranura de registro también puede contener una versión de registro correspondiente. Cada una de las páginas que se corresponden con la base de datos 221 se identifica y se actualiza de tal modo que la lista de las ranuras de registro libres está vacía. A continuación, el procedimiento 800 se lleva a cabo para cada ranura de registro de cada página que se corresponde con la base de datos.

Se accede al límite de comienzo de intervalo válido (por ejemplo, el campo 301) para la ranura de registro correspondiente (el acto 811). A continuación, el flujo se bifurca dependiendo de si el límite de comienzo de intervalo válido indica que la ranura de registro no es parte de una transacción (“Cero” en el bloque de decisión 812), es parte de un registro no confirmado (“TID”, *transaction ID*, en el bloque de decisión 812), u otra cosa (“TS”, *timestamp*, en el bloque de decisión 812).

Se puede concluir que el límite de comienzo de intervalo válido no es en absoluto parte de transacción alguna si, por ejemplo, el límite de comienzo de intervalo válido indica un cero (“Cero” en el bloque de decisión 812). En ese caso, la ranura de registro se añade a la lista libre de la página correspondiente (el acto 813). En particular, si se emplea la forma de realización de estado original, la ranura de registro se llena con el patrón de segundo plano y se proporciona al sistema persistente. A continuación, la ranura de registro se añade a la lista libre de la página correspondiente.

Se puede concluir que el límite de comienzo de intervalo válido es parte de una transacción no confirmada si, por ejemplo, el límite de comienzo de intervalo válido indica un ID de transacción (“TID”, *transaction ID*, en el bloque de decisión 812). Por consiguiente, la versión de registro se puede liberar (el acto 820). Por ejemplo, el límite de comienzo de intervalo válido (por ejemplo, el campo 301) se puede restablecer a cero. A continuación, la ranura de registro se puede llenar con el patrón de segundo plano y proporcionarse al sistema persistente si se emplea la forma de realización de estado original y, a continuación de lo anterior (en la forma de realización o bien de suma de comprobación o bien de estado original), se añade a la lista de ranuras de registro libres de la página correspondiente (el acto 813). Cuando se proporciona una ranura de registro a un sistema persistente, solo es necesario que se sobrescriban las unidades atómicas de la ranura de registro que no contienen ya el patrón de segundo plano. Además, en la presente forma de realización de estado original solo es necesario proporcionar una línea de memoria caché al sistema persistente si el mismo contiene al menos una unidad atómica que se ha sobrescrito. La mayor parte de las ranuras de registro libres pueden llegar al acto 813 por medio de la ruta de “Cero” fuera del bloque de decisión 812 y, por lo tanto, es probable que puedan tener ya el patrón de segundo plano. Por consiguiente, es probable que leer antes de escribir sea una buena optimización para las mismas. Por otro lado, es mucho menos probable que las ranuras de registro libres que llegan al acto 813 desde el acto 820 contengan el patrón de segundo plano, por lo tanto puede ser más eficiente sobrescribir las mismas con el patrón de segundo plano de forma incondicional, y escribir, sin asignación de escritura, en la memoria caché.

El límite de comienzo de intervalo válido (por ejemplo, el campo 301) también se puede evaluar para concluir que el límite de fin de intervalo válido (por ejemplo, el campo 302) se debería evaluar debido a que el límite de comienzo incluye una marca de tiempo (“TS”, *timestamp*, en el bloque de decisión 812). En el presente caso, la ranura de registro se corresponde con una versión de registro que se creó a través de una transacción confirmada.

En el presente caso, se accede al límite de fin de intervalo válido para la ranura de registro (el acto 815) y, a continuación, el flujo de procesamiento se bifurca dependiendo de la naturaleza del límite de fin de intervalo válido (el bloque de decisión 816). Por ejemplo, si el límite de fin de intervalo válido indica infinito (“Infinito” en el bloque de decisión 816), esto quiere decir que la ranura de registro se corresponde con una versión de registro que es actual. Por consiguiente, la recuperación identifica a qué tabla de la base de datos pertenece la versión de registro actual (el acto 817), y añade la versión de registro a la tabla identificada (el acto 818), incluyendo insertar la versión en todos los índices que se asocian con la tabla.

Si el límite de fin de intervalo válido indica un ID de transacción (“TID”, *transaction ID*, en el bloque de decisión 816), esto quiere decir que una transacción intentó hacer antigua la versión de registro correspondiente, pero que la transacción no se confirmó. Por consiguiente, el límite de fin de intervalo válido se actualiza (el acto 819) para hacer que la versión de registro sea actual de nuevo (al hacer el límite de fin de intervalo válido igual a infinito) y la actualización se proporciona al sistema persistente. Además, la recuperación identifica a qué tabla de la base de datos pertenece la versión de registro actual (el acto 817), y añade la versión de registro a la tabla identificada (el acto 818), incluyendo insertar la versión en todos los índices que se asocian con la tabla.

Si el límite de fin de intervalo válido indica una marca de tiempo (“TS”, *timestamp*, en el bloque de decisión 816), esto quiere decir que la versión se hizo antigua por medio de una transacción confirmada. Por consiguiente, la versión de registro se puede liberar (el acto 820). Por ejemplo, el límite de comienzo de intervalo válido se puede restablecer a cero, indicando la disponibilidad de la ranura de registro para su uso en la creación de una versión de registro futura. Las actualizaciones a la ranura de registro se pueden proporcionar al sistema persistente y, a continuación, la ranura de registro se puede añadir a la lista libre de ranuras de registro (el acto 813).

Por consiguiente, los principios que se describen en el presente documento proporcionan un mecanismo eficaz para llevar a cabo transacciones en una base de datos que se hace persistir en un sistema persistente, y la recuperación de un fallo de sistema en el contexto de las mismas.

La presente invención se puede materializar en otras formas específicas sin apartarse de su espíritu o características esenciales. Las formas de realización descritas se han de considerar en todos los aspectos solo como ilustrativas y no restrictivas. El alcance de la invención, por lo tanto, se indica por medio de las formas de realización adjuntas en lugar de por medio de la descripción anterior. Todos los cambios que entren dentro del significado y el ámbito de equivalencia de las reivindicaciones se han de considerar englobados dentro de su alcance.

REIVINDICACIONES

1. Un procedimiento para llevar a cabo una transacción en una base de datos (221) que incluye una colección de registros en un sistema persistente (220A, 220B) que está acoplado con uno o más procesadores (102; 210A, 210B) por medio de un bus (230), en el que el sistema persistente sirve como al menos parte de una memoria principal, en el que la transacción pone en práctica un control de múltiples versiones en el que un registro no se actualiza en su sitio, comprendiendo el procedimiento la siguiente secuencia de etapas:

para cada versión de registro que se crea por medio de la transacción actual, llevar a cabo lo siguiente:
antes de almacenar una versión de registro nueva en una ranura de registro vacía en el sistema persistente, llenar la ranura de registro con un patrón de bits de segundo plano previamente determinado y hacer persistir el patrón; cuando la versión de registro se escribe en la ranura de registro vacía, comprobar si el patrón de segundo plano se pone de manifiesto; almacenar en una memoria intermedia de registro para la transacción una información de procesamiento de post confirmación para el sistema persistente en el que la información de procesamiento de post confirmación incluye al menos un marcador de confirmación para una transacción actual, un puntero para cada versión de registro que se crea por medio de la transacción actual y para cada versión de registro que se actualiza por medio de la transacción actual, y una lista de mapas de bits, como máximo uno para cada versión de registro recién creada, en el que cada bit en los mapas de bits se corresponde con una unidad en la ranura de registro vacía en la que se almacenó la versión de registro nueva que se puede hacer persistir de forma atómica en el sistema persistente, y en el que un bit se establece para cualquier unidad atómica que resulta que contiene el patrón de segundo plano y, de lo contrario, no se establece;
en un tiempo de recuperación, comprobar las unidades atómicas de las versiones de registro nuevas de la transacción, en el que si una unidad atómica no contiene el patrón de segundo plano, concluir que la unidad atómica se hizo persistir, y si la unidad atómica contiene el patrón de segundo plano, comprobar un bit correspondiente del mapa de bits de la versión de registro, y si el bit se establece, concluir que la unidad atómica es correcta y, de lo contrario, concluir que la unidad atómica no se hizo persistir en la base de datos.

2. El procedimiento de acuerdo con la reivindicación 1, que comprende adicionalmente:

para cada versión de registro que se crea por medio de la transacción actual, proporcionar la versión de registro al sistema persistente; y
para cada versión de registro que se actualiza por medio de la transacción actual, proporcionar al menos una porción de la versión de registro al sistema persistente.

3. Un sistema (100; 200) que comprende:

uno o más procesadores (102; 210A, 210B);
una base de datos (221) que incluye una colección de registros en un sistema persistente (220A, 220B), en el que el sistema persistente sirve como al menos parte de una memoria principal;
un bus (230) que acopla los uno o más procesadores con el sistema persistente;
un módulo de transacción (201) que está configurado para llevar a cabo un procedimiento de transacción sobre una colección de registros en el sistema persistente, en el que la transacción pone en práctica un control de múltiples versiones en el que un registro no se actualiza en su sitio, comprendiendo el procedimiento:

para cada versión de registro que se crea por medio de la transacción actual, llevar a cabo lo siguiente:

antes de almacenar una versión de registro nueva en una ranura de registro vacía en el sistema persistente, llenar la ranura de registro con un patrón de bits de segundo plano previamente determinado y hacer persistir el patrón; cuando la versión de registro se escribe en la ranura de registro vacía, comprobar si el patrón de segundo plano se pone de manifiesto; almacenar en una memoria intermedia de registro para la transacción una información de procesamiento de post confirmación para el sistema persistente, en el que la información de procesamiento de post confirmación incluye al menos un marcador de confirmación para una transacción actual, un puntero para cada versión de registro que se crea por medio de la transacción actual y para cada versión de registro que se actualiza por medio de la transacción actual, y una lista de mapas de bits, como máximo uno para cada versión de registro recién creada, en el que cada bit en los mapas de bits se corresponde con una unidad en la ranura de registro vacía en la que se almacenó la versión de registro nueva que se puede hacer persistir de forma atómica en el sistema persistente, y en el que un bit se establece para cualquier unidad atómica que resulta que contiene el patrón de segundo plano y, de lo contrario, no se establece;

en un tiempo de recuperación, comprobar las unidades atómicas de las versiones de registro nuevas de la transacción, en el que si una unidad atómica no contiene el patrón de segundo plano, concluir que la unidad atómica se hizo persistir, y si la unidad atómica contiene el patrón de segundo plano, comprobar un bit correspondiente del mapa de bits de la versión de registro, y si el bit se establece, concluir que la unidad atómica es correcta y, de lo contrario, concluir que la unidad atómica no se hizo persistir en la base de datos.

4. El sistema de acuerdo con la reivindicación 3, que está adaptado adicionalmente para llevar a cabo las etapas de:

para cada versión de registro que se crea por medio de la transacción actual, proporcionar la versión de registro al sistema persistente; y
para cada versión de registro que se actualiza por medio de la transacción actual, proporcionar al menos una porción de la versión de registro al sistema persistente.

5. Una memoria informática física y tangible que tiene en la misma unas instrucciones ejecutables por ordenador que, cuando son ejecutadas por un procesador, dan lugar a que el procesador lleve a cabo el procedimiento de una cualquiera de las reivindicaciones 1 a 2.

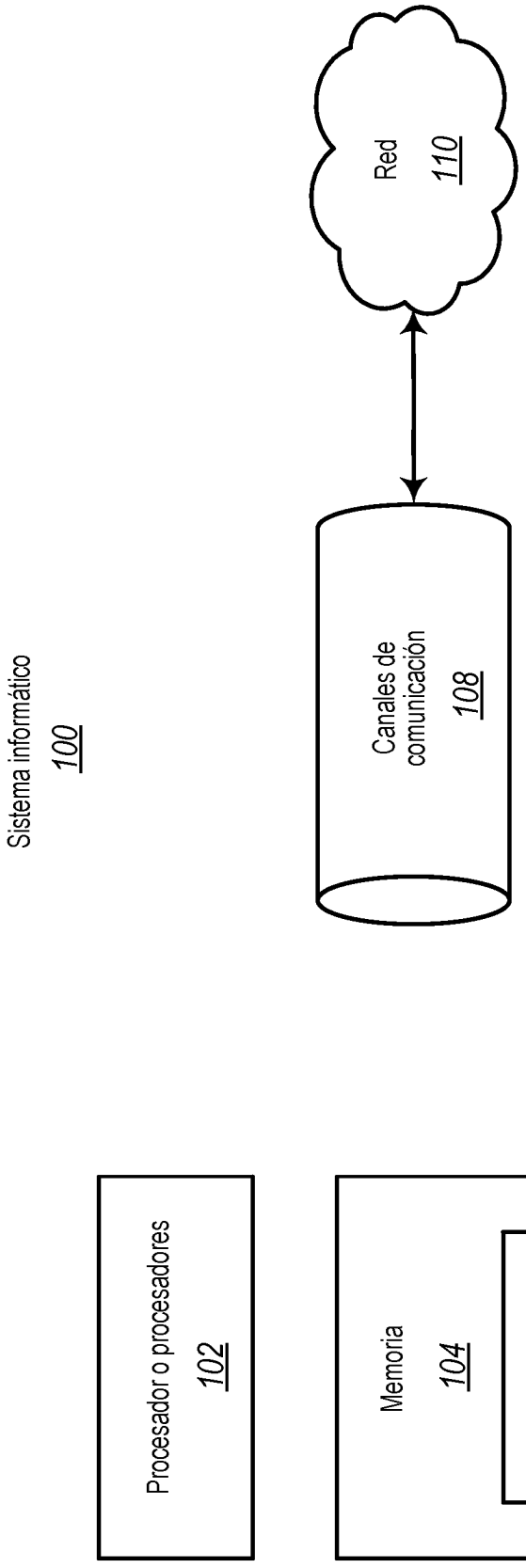
Sistema informático
100
Red
110
Canales de comunicación
108
Procesador o procesadores
102
Memoria
104
Volátil
No volátil

FIG. 1

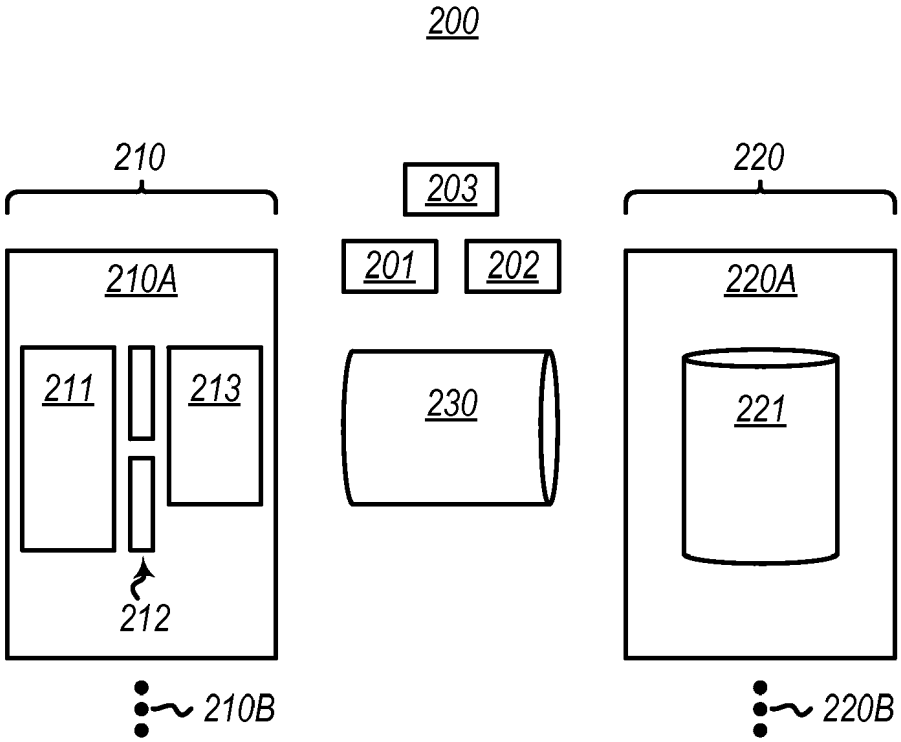
200
210
203
220
210A
201
202
220A
211
213
230
221
212
210B
220B

FIG. 2

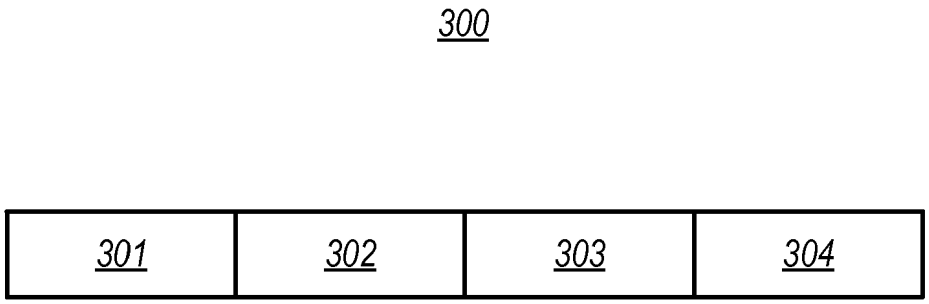
300
301
302
303
304

FIG. 3

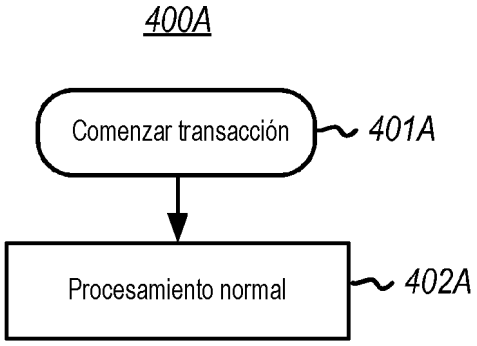

FIG. 4A

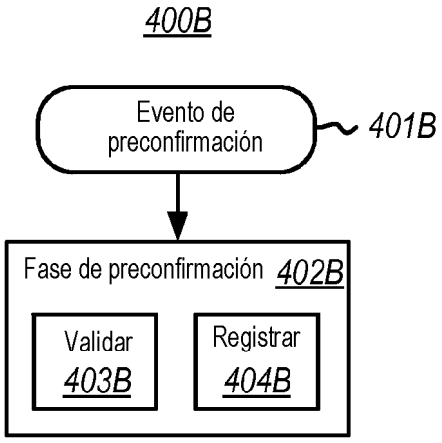

FIG. 4B

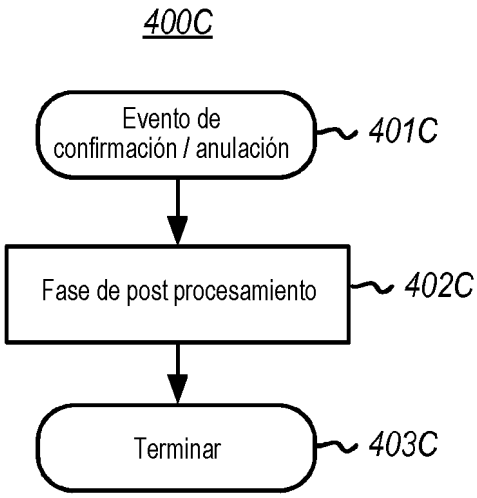

FIG. 4C

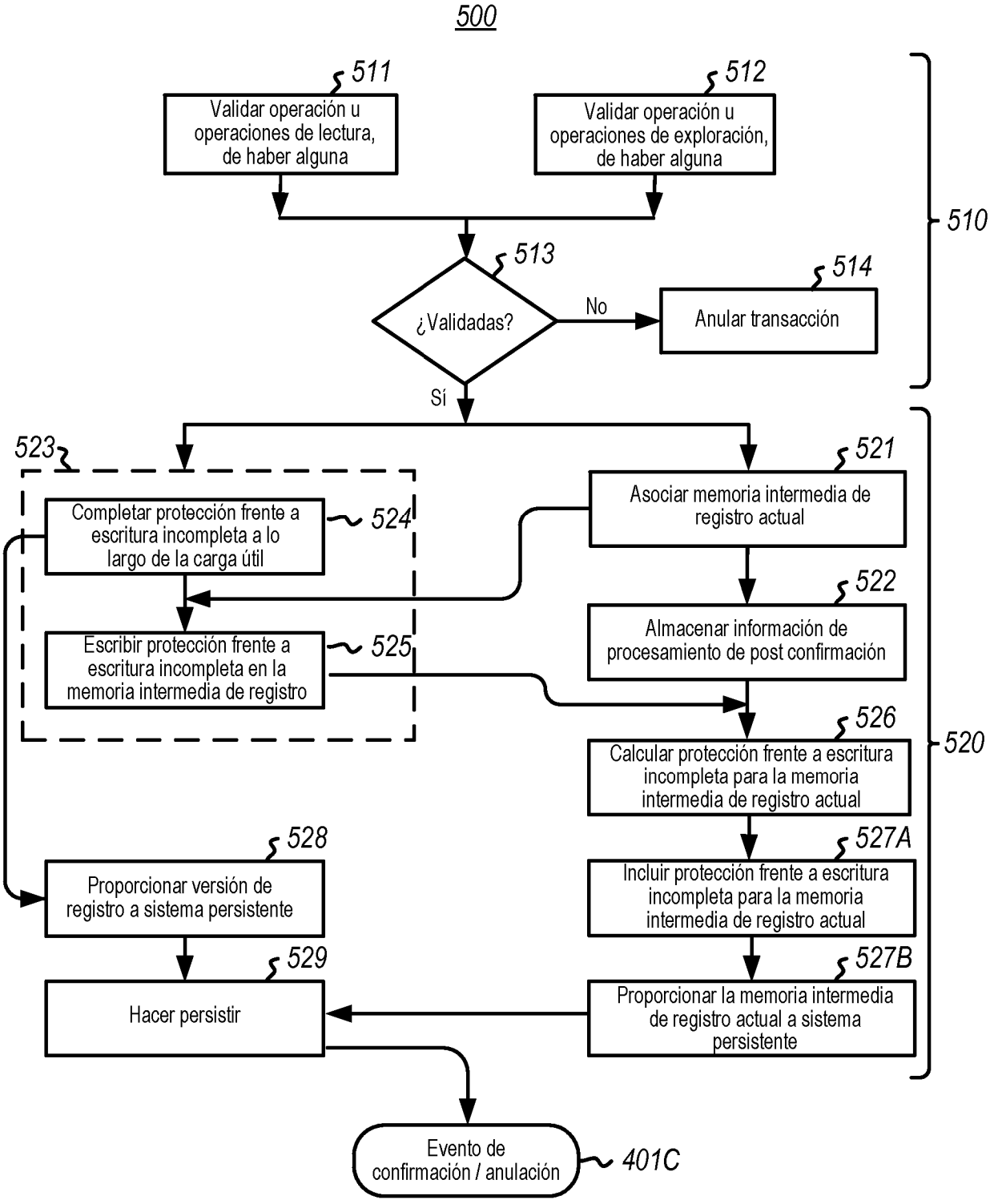
500
511
Validar operación u operaciones de lectura, de haber alguna
512
Validar operación u operaciones de exploración, de haber alguna
510
513
¿Validadas?
No
514
Anular transacción
Sí
523
521
Asociar memoria intermedia de registro actual
Completar protección frente a escritura incompleta a lo largo de la carga útil
524
522
Almacenar información de procesamiento de post confirmación
Escribir protección frente a escritura incompleta en la memoria intermedia de registro
525
526
Calcular protección frente a escritura incompleta para la memoria intermedia de registro actual
520
528
527A
Proporcionar versión de registro a sistema persistente
Incluir protección frente a escritura incompleta para la memoria intermedia de registro actual
529
527B
Hacer persistir
Proporcionar la memoria intermedia de registro actual a sistema persistente
Evento de confirmación / anulación
401C

FIG. 5

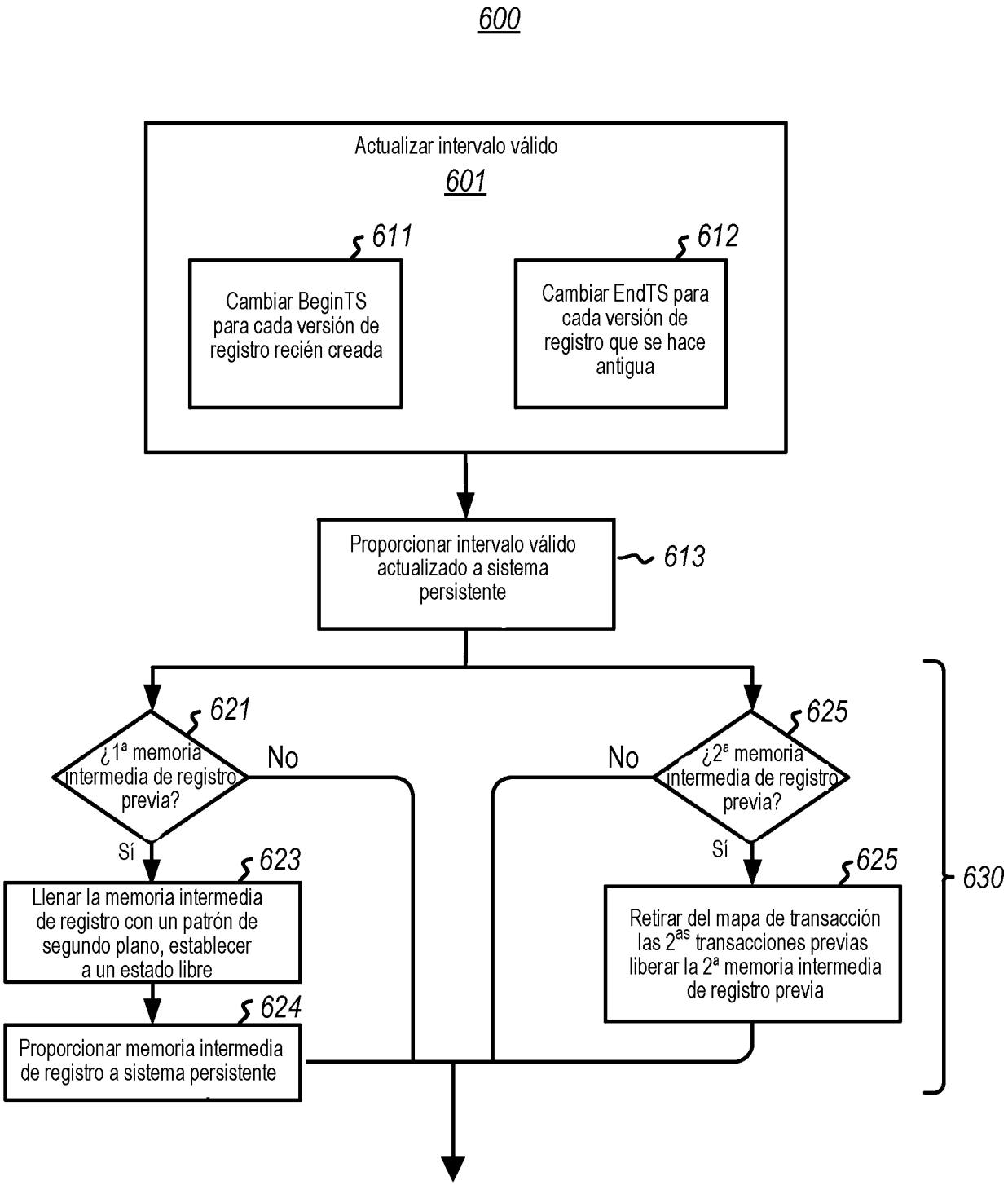
600
Actualizar intervalo válido
601
611
Cambiar BeginTS para cada versión de registro recién creada
612
Cambiar EndTS para cada versión de registro que se hace antigua
Proporcionar intervalo válido actualizado a sistema persistente
613
621
¿1ª memoria intermedia de registro previa?
No
Sí
623
Llenar la memoria intermedia de registro con un patrón de segundo plano, establecer a un estado libre
624
Proporcionar memoria intermedia de registro a sistema persistente
625
¿2ª memoria intermedia de registro previa?
No
Sí
625
Retirar del mapa de transacción las 2as transacciones previas liberar la 2ª memoria intermedia de registro previa
630

FIG. 6

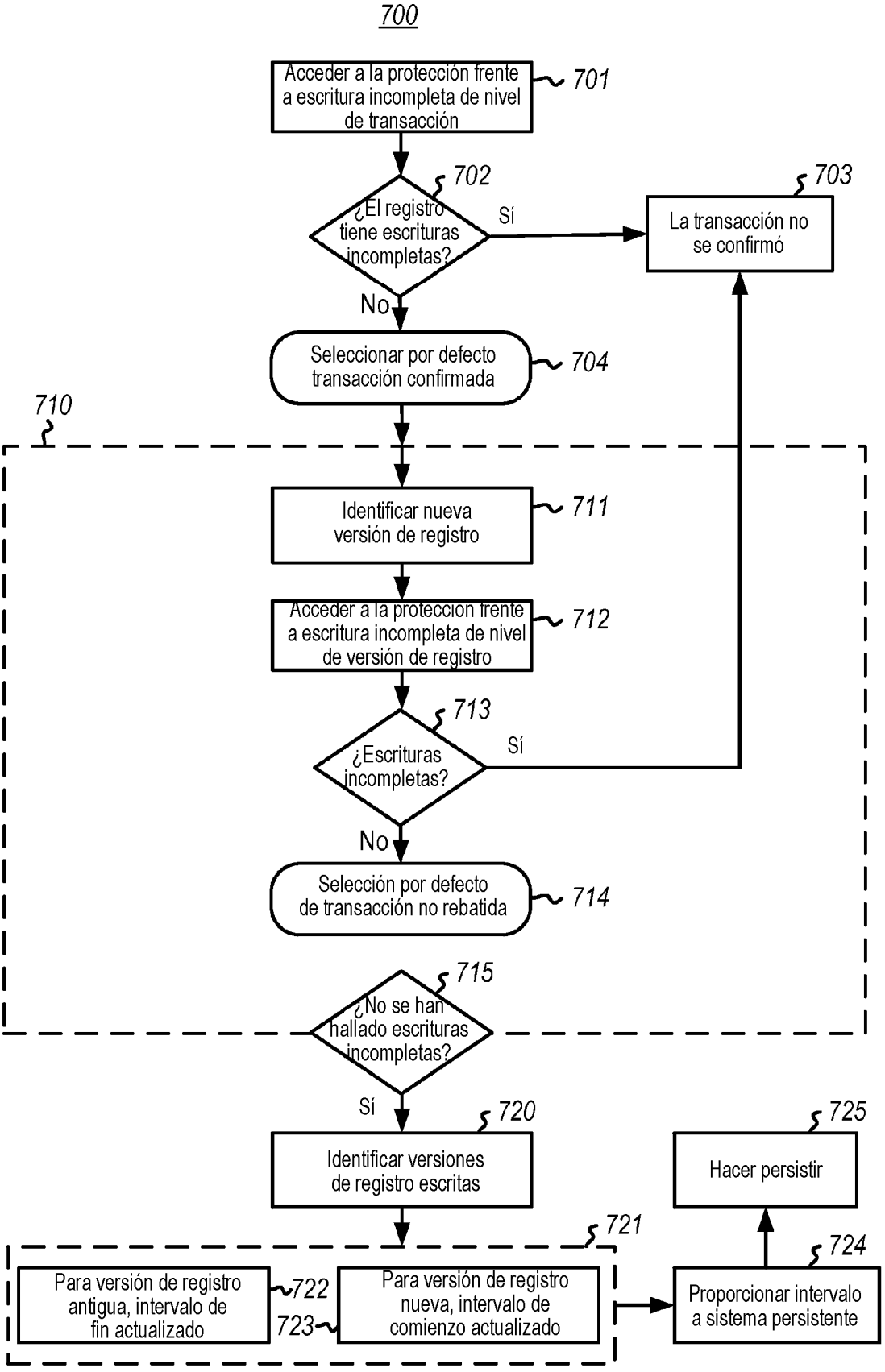
700
Acceder a la protección frente a escritura incompleta de nivel de transacción
701
¿El registro tiene escrituras incompletas?
702
Sí
La transacción no se confirmó
703
No
Seleccionar por defecto transacción confirmada
704
710
Identificar nueva versión de registro
711
Acceder a la proteccion frente a escritura incompleta de nivel de versión de registro
712
¿Escrituras incompletas?
713
Sí
No
Selección por defecto de transacción no rebatida
714
¿No se han hallado escrituras incompletas?
715
Sí
Identificar versiones de registro escritas
720
721
Para versión de registro antigua, intervalo de fin actualizado
722
Para versión de registro nueva, intervalo de comienzo actualizado
723
Proporcionar intervalo a sistema persistente
724
Hacer persistir
725

FIG 7

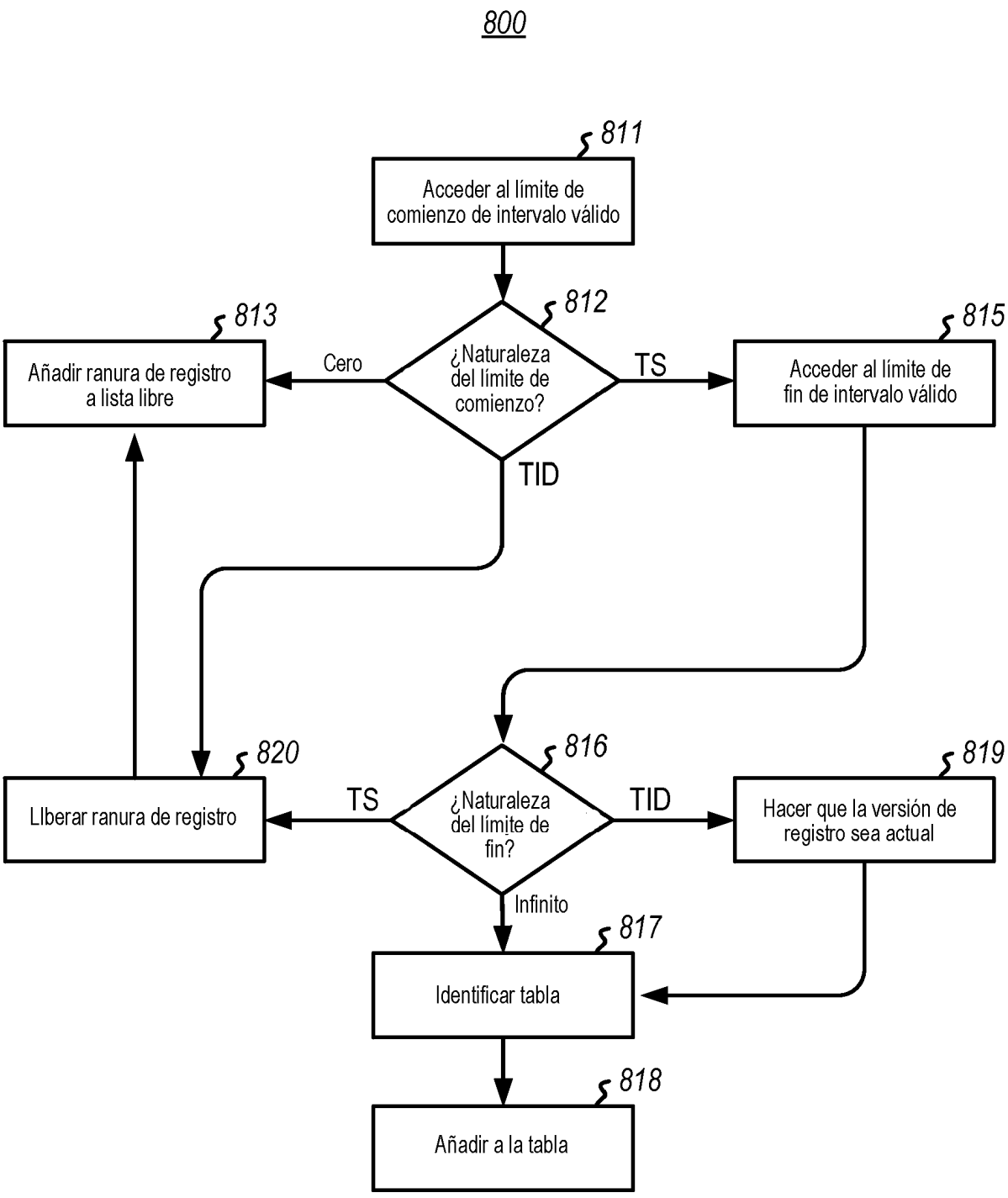
800
811
Acceder al límite de comienzo de intervalo válido
812
¿Naturaleza del límite de comienzo?
Cero
TS
TID
813
Añadir ranura de registro a lista libre
815
Acceder al límite de fin de intervalo válido
816
¿Naturaleza del límite de fin?
TS
TID
Infinito
820
Llberar ranura de registro
819
Hacer que la versión de registro sea actual
817
Identificar tabla
818
Añadir a la tabla

FIG. 8