(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 363 796**

(51) Int. Cl.:
H04L 29/08 (2006.01)
G06F 17/30 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Número de solicitud europea: **06793877 .9**
(96) Fecha de presentación : **27.09.2006**
(97) Número de publicación de la solicitud: **1932322**
(97) Fecha de publicación de la solicitud: **18.06.2008**

(54) Título: **Sistema y método para mantener la coherencia de un contenido "caché" en un sistema de software de niveles múltiples destinado a constituir interfaz entre grandes bases de datos.**

(30) Prioridad: **03.10.2005 EP 05109147**
05.10.2005 US 723445 P

(45) Fecha de publicación de la mención BOPI:
16.08.2011

(45) Fecha de la publicación del folleto de la patente:
16.08.2011

(73) Titular/es: **AMADEUS S.A.S.**
485 Route du Pin Montard, Sophia Antipolis
06410 Biot, FR

(72) Inventor/es: **Janin, Benoît;**
Gole, Rémy;
Isnardy, Luc;
Daniello, Rudy y
Rubenstein, Wayne

(74) Agente: **Isern Jara, Jorge**

DESCRIPCIÓN

Sistema y método para mantener la coherencia de un contenido “caché” en un sistema de software de niveles múltiples destinado a constituir interfaz entre grandes bases de datos.

SECTOR DE LA INVENCIÓN

La presente invención se refiere de manera general a arquitecturas de software de niveles múltiples de cliente/servidor, y más específicamente, a un método y a un sistema para mantener la coherencia entre el contenido de cachés que están implementadas en aparatos de nivel frontal o primer nivel y nivel medio para mejorar el rendimiento global.

ANTECEDENTES DE LA INVENCIÓN.

El modelo cliente/servidor que ha surgido a finales de los años 1980 es una arquitectura de software versátil y modular que ha estado destinada a mejorar la capacidad de utilización, flexibilidad, interoperabilidad y escalabilidad en comparación con la utilización de ordenadores principales centralizados con división de tiempo, que era lo habitual en aquel momento. Un cliente es un solicitante de servicios y el servidor un proveedor de dichos servicios. Dependiendo de la configuración del software, un mismo aparato puede ser tanto cliente como servidor.

La arquitectura cliente/servidor ha sustituido progresivamente la arquitectura de software anteriormente conocida con ordenadores principales (“mainframe”) en los que toda la inteligencia se encontraba en el ordenador central y en el que los usuarios interaccionaban con el ordenador central a través de un terminal no inteligente (“dumb terminal”) que solamente capta pulsaciones de teclado y envía esta información al ordenador principal. Una limitación bien conocida de las arquitecturas de software de ordenador principal es que no soportan fácilmente interfaces de usuario de gráficos (GUI) o el acceso a bases de datos múltiples desde lugares geográficos dispersos. Los ordenadores principales se utilizan, no obstante, todavía como potentes servidores en arquitecturas distribuidas de cliente/servidor.

La arquitectura cliente/servidor ha introducido un servidor de base de datos que actúa como servidor de archivo. En esta arquitectura, las solicitudes del usuario son contestadas utilizando directamente un sistema de gestión de base de datos relacional o RDBMS. El tráfico de la red se reduce al facilitar una respuesta a la petición en vez de transferir siempre los archivos completos. También mejora la actualización de usuarios múltiples a través de un elemento del proceso (“front end”) GUI a una base de datos compartida. En arquitecturas cliente/servidor se utilizan de manera típica las afirmaciones estructuradas en lenguaje de interrogación (SQL) para el intercambio de datos entre clientes y servidores.

Con la arquitectura cliente/servidor de dos niveles (100) mostrada en la figura 1, el interfaz del sistema de usuario está situado en el entorno del ordenador de sobremesa (102) del usuario y los servicios de gestión de la base de datos se encuentran en un servidor (104) capaz de dar servicio a muchos clientes. La gestión del proceso está dividida entre el entorno del interfaz del sistema de usuario y el entorno del servidor de gestión de la base de datos. Todas las combinaciones de topologías, incluyendo el interfaz de clientes único/múltiples, servidores únicos/múltiples (no mostrado), muy frecuentemente en una red de área local o LAN (108), son evidentemente posibles.

En la arquitectura tradicional de dos niveles, el primer nivel, el cliente (102), posee el interfaz de usuario, la lógica principal de negocio y de proceso de datos. Acepta y comprueba la sintaxis de la entrada de usuario, procesa la lógica de aplicación, genera peticiones de bases de datos, las transmite al servidor y pasa la respuesta en retorno a tareas del usuario. El segundo nivel, el servidor de la base de datos (104), acepta y procesa peticiones de base de datos de clientes, comprueba autorizaciones, asegura que no se violen las limitaciones de integridad, lleva a cabo proceso de peticiones/actualización y transmite repuestas al cliente. También mantiene el catálogo del sistema, proporciona acceso simultáneo a la base de datos y lleva a cabo control de recuperación.

La arquitectura cliente/servidor de dos niveles se ha demostrado como una solución satisfactoria para el trabajo con ordenadores distribuido cuando los grupos de trabajo no superan 100 personas que interaccionan en un LAN simultáneamente. No obstante, cuando el número de usuarios aumenta, el rendimiento empieza a deteriorarse como resultado de que el servidor mantiene una conexión mediante mensajes de “mantenimiento vivo” (“keepalive”) con cada cliente, aunque no se esté realizando trabajo. Una segunda limitación de la arquitectura de dos niveles es que la implementación de los servicios de gestión de proceso, utilizando procesos de bases de datos, propiedad del vendedor, restringe la flexibilidad y selección de RDBMS para las aplicaciones. Asimismo, las implementaciones de la arquitectura de dos niveles han demostrado una flexibilidad limitada en el desplazamiento (reparto) de la funcionalidad de un programa del servidor a otro.

Posteriormente, aparecieron en los años 1990 la arquitectura de tres niveles (120) y variantes de niveles múltiples para superar las limitaciones anteriores. En la arquitectura de tres niveles se añadió un nivel intermedio (126) entre

el interfaz del sistema de usuario del entorno del cliente (122) y el entorno (124) del servidor de gestión de la base de datos. Si bien existen múltiples maneras de implementar esta arquitectura y el nivel intermedio, este último tiene frecuentemente la función de ordenación o formación de colas, ejecución de aplicación y disposición de la base de datos. De manera típica, un cliente facilita su petición al nivel intermedio y se desconecta porque el nivel intermedio debe tener acceso a los datos, y devuelve la respuesta al cliente. Además, la capa intermedia añade programación y priorización para el trabajo que se está realizando.

En la variante indicada anteriormente de arquitectura de tres niveles, el cliente, es decir, el primer nivel, puede tener que llevar a cabo solamente el interfaz de usuario, es decir, validar las entradas; en cuyo caso el nivel intermedio soporta toda la lógica de negocio y realiza el proceso de datos, mientras que el servidor, es decir, el tercer nivel, lleva a cabo la validación de datos y controla el acceso a la base de datos.

Los documentos US-A1-2003 0188 106 y US-B1-6934717 dan a conocer arquitecturas de niveles múltiples con cachés de replicación para los datos contenidos en una base de datos principal o maestra, situada en el programa de proceso del sistema (“backend”).

La arquitectura de tres niveles cliente/servidor se ha demostrado que mejora el rendimiento para grupos con un gran número de usuarios (de manera típica, hasta mil, es decir, diez veces el caso de dos niveles) y mejora la flexibilidad en comparación con el sistema de dos niveles, especialmente, a causa de que el código de aplicación no tiene que ser compartido entre capas. La arquitectura cliente/servidor de tres niveles tiene como resultado un entorno que es considerablemente más escalable que la arquitectura de dos niveles con conexión directa del cliente al servidor. Proporciona la capacidad de actualizar múltiples RDBMS distintos en una sola transacción y puede conectarse a una serie de fuentes de datos, incluyendo archivos planos (“flat files”), sistemas de gestión de base de datos no relacionales, y asimismo a ordenadores principales que en la actualidad sirven frecuentemente como poderosos servidores de la base de datos. Las arquitecturas de tres niveles y de niveles múltiples se adaptan, por lo tanto, de manera óptima a entornos grandes de cliente/servidor distribuidos. Por ejemplo, los que tienen que desplegar las compañías de reservas aéreas para servir a sus clientes, es decir, las agencias de viajes en todo el mundo y en los que se requieren recursos compartidos, tal vez como base de datos heterogéneas (es decir, las bases de datos de tarifas y de disponibilidad de las compañías aéreas) y normas de proceso.

El hecho de que los centros de datos de niveles múltiples han pasado a ser una exigencia básica para proporcionar los mencionados servicios, reduciendo la carga de cálculo de los ordenadores y las comunicaciones, es crucial para mejorar adicionalmente el rendimiento y la escalabilidad. El disponer en memoria caché contenidos dinámicos en varios niveles de un centro de datos de niveles múltiples es un método bien conocido para reducir las cargas de cálculo de ordenador y de comunicación, de manera que se pueda servir a un mayor número de clientes simultáneamente, puesto que en caso de acceso (“hit”), los datos no tienen que ser buscados nuevamente desde un nivel situado por encima. No obstante, la disposición en caché en niveles intermedios y primeros (“front tiers”) tiene sus propias dificultades. La consistencia de la memoria caché y la coherencia de la misma pasan a ser temas que deben ser tratados de la manera correspondiente. Especialmente, para reservas de líneas aéreas en las que no son aceptables valores obsoletos de la disponibilidad de las líneas aéreas, es esencial una fuerte consistencia y coherencia.

OBJETO DE LA INVENCIÓN

Por lo tanto, un objetivo amplio de la invención consiste en dar a conocer un método y sistema para mantener la coherencia del contenido dinámico (caché) en arquitecturas de software de niveles múltiples.

Es un objetivo más específico de la invención que ello se adapte a arquitecturas de niveles múltiples tales como las utilizadas para los sistemas de reservas de líneas aéreas, y que se caracterizan por un nivel muy elevado de transacciones desde el lado de los clientes y actualizaciones muy frecuentes de las bases de datos de tarifas y de disponibilidad, proporcionadas por las compañías aéreas y otros proveedores y servicios similares.

Otros objetivos, características y ventajas de la presente invención quedarán evidentes para los técnicos en la materia a partir de la siguiente descripción, haciendo referencia a los dibujos adjuntos. Se plantea que, cualesquiera ventajas adicionales queden incorporadas a esta descripción.

RESUMEN DE LA INVENCIÓN

Se describe un método y sistema para mantener la coherencia de un contenido de caché en una arquitectura de software de múltiples niveles. Ello incluye un primer nivel de servidores satélite, cada uno de los cuales opera una caché local y un nivel intermedio de servidores centrales, cada uno de los cuales utiliza una caché central. Los servidores centrales se interconectan con bases de datos a través de servidores de bases de datos para recuperar los elementos de datos utilizados para construir objetos. Una vez construidos, los objetos reciben la atribución de un tiempo de vigencia (“time-to-live” (TTL)) y se almacenan en cachés centrales, y a continuación se envían a los servidores satélite donde quedan almacenados en cachés locales antes de ser suministrados a las aplicaciones de

software que los han solicitado. Las peticiones de servidores satélite son equilibrados en cuanto a carga sobre la totalidad de servidores centrales disponibles. Un servidor central es seleccionado para cada petición de gestión. Un objeto de nueva construcción es replicado en todos los demás servidores centrales desde el servidor central seleccionado. Un objeto es solicitado de la caché central seleccionada siempre que falte o esté anticuado en la caché local. La construcción de un objeto requerido es puesta en marcha en el servidor central seleccionado siempre que falte en la caché central seleccionada. La construcción es eliminada si el objeto requerido se encuentra ya presente y no está anticuado en la caché central. Un servidor central está diseñado como servidor central principal y todos los demás son servidores centrales de seguridad o refuerzo. La construcción es puesta en marcha en el servidor central desde un gestionador de invalidación, siempre que el objeto solicitado resulta anticuado. El TTL del objeto que ha resultado anticuado se dispone en un valor anticuado se dispone en un valor bajo antes de ser enviado al servidor satélite solicitante. Los objetos almacenados en cachés centrales y en cachés locales son invalidados tan pronto como, por lo menos, un elemento de datos utilizado para su construcción ha sido modificado en las bases de datos, lo que determina cuales objetos han sido afectados, enviando a continuación instrucciones de invalidación a todos los gestionadores de invalidación del servidor central. Éstos invalidan los objetos correspondientes en las cachés centrales, y a continuación propagan las instrucciones de invalidación a todas las cachés locales que, a su vez, invalidan y anulan los correspondientes objetos en las cachés locales. Los objetos invalidados en cachés centrales son borrados o reconstruidos. En este último caso, los objetos reconstruidos son replicados en todas las cachés centrales de refuerzo.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

La figura 1 explica la técnica anterior, es decir, las arquitecturas de software de 2 niveles en comparación con las de niveles múltiples.

La figura 2 muestra la arquitectura de software global de niveles múltiples a la que se aplica de forma óptima la invención.

La figura 3 describe los diferentes casos de búsqueda de objeto que resultan del proceso de peticiones de usuario final por un programa de aplicación y la construcción de objetos en servidores centrales.

La figura 4 explica el envejecimiento de objetos en cachés local y central.

La figura 5 describe la invalidación de objetos en cachés cuando se modifican elementos de datos en bases de datos.

La figura 6 explica la reconstrucción de objetos invalidados.

La figura 7 describe el almacenamiento, en una segunda memoria de objetos, no anticuados que deben ser retirados de las cachés centrales.

DESCRIPCIÓN DETALLADA

La siguiente descripción detallada de la invención se refiera a los dibujos adjuntos. La descripción incluye realizaciones a título de ejemplo.

La **figura 2** describe la arquitectura de software de niveles múltiples global a la que es aplicable en mejor medida la presente invención.

El nivel superior (200) es la fuente última de datos en la que, como mínimo, un servidor de base de datos (202) se interconecta de manera general con bases de datos múltiples (204), tales como las bases de datos de disponibilidad y tarifas proporcionadas por las líneas aéreas y otros proveedores de dichos servicios de todo el mundo. Estas bases de datos son actualizadas frecuentemente por sus usuarios, es decir, los que tienen la responsabilidad de su actualización y de mantener su contenido (206), frecuentemente a través de una combinación de redes privadas y públicas (208) que comprenden Internet. De modo global, en base diaria, se pueden registrar millones de transacciones comportando grandes volúmenes de datos.

El nivel intermedio (210) se ha mostrado en este caso formado por dos servidores a los que se hará referencia a continuación como servidor de datos central o CDS. De manera típica, a efectos de redundancia existe un servidor principal (212) y un servidor de apoyo (214). No obstante, son posibles cualesquiera configuraciones entre uno (sin redundancia) y muchos servidores. Asimismo, un servidor central puede tener que ser ocasionalmente configurado como servidor autónomo, por ejemplo, facilitando el mantenimiento del sistema o porque el CDS principal previsto se encuentra fuera de funcionamiento. Además, un servidor de soporte designado puede tener que funcionar temporalmente como servidor principal en caso necesario. Esto se explica adicionalmente en la siguiente descripción de la invención.

El disponer de múltiples servidores en el nivel intermedio es una práctica habitual con esta arquitectura. Puede existir más de un servidor redundante desempeñando el mismo papel que el servidor principal o servidores especializados que utilizan diferentes programas de aplicación. Cuando se requiere mayor potencia de proceso, una forma de conseguir la escalabilidad consiste en añadir servidores intermedios de manera que se pueden manejar más transacciones de clientes a través del equilibrado de carga de las peticiones del primer nivel (220) sobre un mayor número de servidores del nivel intermedio. Por lo tanto, cada CDS tiene su propio caché (216, 218) al cual se hace referencia como cachés centrales que deben mantener en todo momento contenidos coherentes.

En el tipo de aplicaciones consideradas por la invención, las entidades de datos presentes en las cachés se designan de manera amplia como objetos de software o simplemente objetos en la siguiente descripción. En general, muchos elementos de datos obtenidos de las bases de datos mediante peticiones, necesitan ser reunidos para crearlos. Por lo tanto, un objeto según la invención es, por ejemplo, una tarifa de viaje específica que ha sido construida por un CDS a partir de elementos de datos obtenidos de las bases de datos (204) a través del servidor de la base de datos (202). Si, tal como se ha explicado en la sección de antecedentes, la base de datos es una base de datos relacional, ello ha sido conseguido emitiendo, como mínimo, una petición SQL y en general muchas más, a las bases de datos de manera que todos los elementos de datos necesarios para construir el objeto, por ejemplo, las tarifas de viaje de este ejemplo específico, se pueden reunir en el CDS. Por lo tanto, los objetos, según la invención, se supone que son objetos elaborados que requieren potencia de proceso y que utilizan ancho de banda de comunicación con el servidor de datos para que puedan ser puestos en forma utilizable. Los objetos pueden permanecer en cachés siempre que la fuente de información utilizada para construirlas no ha sido modificada en las bases de datos. La reconstrucción es costosa dado que consume potencia de proceso y utiliza parte del ancho de banda de comunicación disponible con el servidor de base de datos y sus bases de datos. En lo que se refiere a la coherencia, un objeto específico presente por ejemplo en la caché central del CDS principal (216) debe ser exactamente el mismo que su clon en la caché central del CDS de refuerzo (218) y su contenido debe ser consistente con los elementos de datos de las bases de datos (204) a partir de los que fueron construidos. Esto se explica adicionalmente en la siguiente descripción de la invención.

El primer nivel (220) está realizado a base de una serie de servidores satélite que utilizan aplicaciones de software para sus usuarios finales. En el ejemplo utilizado para ilustrar la invención, éstos son típicamente elementos de asignación de precio o de tarifa. Estas aplicaciones de software pueden ser utilizadas directamente en un servidor satélite de cliente (222) o en servidores independientes satélite (224) del primer nivel, incluyendo grupos de servidores (225) a los que se tiene acceso a su vez desde usuarios en posición remota (226) a través de una red privada o pública (228), por ejemplo, Internet, utilizando navegadores estándar y la aplicación de Internet más extendida: la web a escala mundial o web. En ambos casos, las aplicaciones aprovechan cachés locales (230) esencialmente para reducir la carga de comunicación entre el primer nivel y los CDS del nivel intermedio a los que están conectados todos los servidores satélite. Los objetos que se han explicado anteriormente son llevados, por lo tanto, cuando ello es necesario, también a diferentes cachés locales. Por lo tanto, las aplicaciones locales no tienen que acceder a un CDS o a un servidor de datos si el objeto solicitado se encuentra en su caché local. Esto tiene la ventaja principal de proteger los servidores de la base de datos (202), es decir, impedir que estos servidores reciban grandes cantidades de peticiones de los usuarios finales (226) que de otro modo llegarían a aquellos.

Dado que las aplicaciones de software están diseñadas con el objetivo de conseguir proporciones satisfactorias de accesos (“hit”) de la caché, sus rendimientos quedan notablemente mejorados. Además, esto reduce considerablemente la carga de comunicación entre niveles y finalmente permite adaptar muchos usuarios finales en una misma infraestructura. Tal como ya se ha explicado en lo anterior, todo esto supone que se mantiene la coherencia de las cachés que se explica adicionalmente en las figuras siguientes.

La **figura** 3 describe los varios casos de búsqueda de objeto que resultan del proceso de las peticiones del usuario final por un programa de aplicación.

Cuando este programa de aplicación, por ejemplo, un programa de asignación de precio (300), situado en el servidor de primer nivel, necesita un objeto (305) para calcular un precio de viaje, la caché local es interrogada en primer lugar (310). Si existe acierto (“hit”), el objeto es simplemente recogido de la caché local (315). Esta es la forma más efectiva de recoger o recuperar un objeto dado que el proceso conjunto comporta solamente que el servidor utiliza la aplicación.

Cuando el objeto solicitado no se encuentra presente en la caché local, no obstante, se produce un fallo (“miss”) (320). El objeto debe ser traído y se supone en primer lugar que el objeto se encuentra presente en la caché central del CDS principal (340) al que se envía una petición (390) a través de una función de equilibrado de carga que se explica más adelante. Si este es el caso (345), una función de gestión de datos (350) es destinada a recuperarlo de la caché central para enviarlo (330) a la caché local del servidor peticionario donde se almacena. Después de realizar esta tarea, el objeto solicitado puede ser eventualmente suministrado al programa de aplicación. Por lo tanto, la caché local contiene uno o varios objetos (325) que pueden ser utilizados posiblemente para peticiones adicionales.

Cuando el objeto solicitado no se encuentra presente en la caché local (350) y tampoco está presente en la caché central del CDS principal (355), debe ser construido en CDS y almacenado en la caché central. Esto se consigue por el gestionador de datos (350) que ya se ha mencionado que debe recoger de la base de datos (365) todos los elementos de datos (360) necesarios para construir el objeto requerido (355). La construcción del objeto puede ser un proceso complejo que puede requerir la emisión de muchas peticiones (por ejemplo: consultas SQL) a la base de datos. Una vez se ha reunido se lleva a cabo una comprobación completa del objeto para asegurarse de que es realmente utilizable por la aplicación solicitante. Esto incluye una comprobación sintáctica y semántica del lenguaje de descripción del objeto además de una prueba con un código de muestra de aplicación. Si el nuevo objeto pasa esta fase de validación, tal como lo hace normalmente, se almacena en la caché central. Antes de almacenar, se asocia al objeto un tiempo de validez (TTL) o una fecha de expiración, de manera que puede ser retirado cuando ha transcurrido el tiempo, tal como se explica adicionalmente en la figura 4. Después de ello, el nuevo objeto es enviado a la caché local del servidor que lo ha pedido, donde es almacenado en el primer lugar y a continuación es suministrado (375) a la aplicación de software que lo necesita. No obstante, si no es rechazada la validación de un objeto de nueva construcción, éste es rechazado, por lo que no es almacenado en la caché central o enviado a la caché local (357).

Cuando se ha creado un nuevo objeto en el CDS principal, también se replica 380 en el servidor o servidores centrales de refuerzo (342) de manera que el nuevo objeto puede ser recuperado también desde las otras cachés del servidor central. Tal como se ha explicado anteriormente, para permitir la redundancia y escalabilidad, muchos servidores intermedios se pueden encontrar activos simultáneamente a efectos de que las peticiones de los servidores del primer nivel puedan ser equilibradas en cuanto a carga sobre el conjunto de servidores de datos centrales disponible. El equilibrado de carga, esquematizado en la figura 3, que se encuentra más allá del alcance de la invención, no es explicado adicionalmente a parte de mencionar que es una función de equilibrado de carga (385) que envía las peticiones (390) del primer nivel, de acuerdo con una lógica o algoritmo específicos, que pueden ser tan simples como la bien conocida función “round-robin”. En este caso, las peticiones del primer nivel son enviadas sucesivamente a cada servidor central activo de manera secuencial hasta que se alcanza el último punto. La siguiente petición es enviada entonces al servidor central numerado en primer lugar y así sucesivamente. Este simple algoritmo es frecuentemente muy eficaz. No obstante, existen otros más sofisticados tales como los que miden la carga real de cada servidor y gestionan para enviar una petición entrante al que está menos ocupado.

Por lo tanto, la descripción anterior que mencionó que un objeto faltante es requerido de la caché del CDS principal, debe ser corregida. Dependiendo de la decisión realizada por la función de equilibrado de carga (385), el objeto faltante puede ser requerido también desde una caché central de refuerzo dado que los objetos de nueva construcción son replicados (380) en servidores activos. En este caso, si el objeto, no obstante, no se encuentra presente en la caché central de refuerzo seleccionado, debe ser construido, tal como se ha explicado anteriormente, igual que si se hubiera seleccionado el CDS central. Cuando esto ocurre, el refuerzo pasa a ser temporal como un CDS principal, lo que significa que una vez construido, el objeto es replicado desde el refuerzo a todos los CDS activos, de manera que el nuevo objeto pasa a estar disponible en todas las cachés del nivel intermedio. Este aspecto de la invención, según el cual un CDS de refuerzo puede actuar temporalmente como CDS principal, se explica también en la figura 6.

La **figura 4** explica la pérdida de validez o “envejecimiento” de los objetos en las cachés. Tal como se ha mencionado anteriormente, todos los objetos construidos tienen un tiempo de duración (TTL) definido o fecha de expiración asociada a los mismos, de manera que los elementos que están fuera de su tiempo de validez pueden ser eliminados de las cachés y no permanecen para siempre en memoria.

Cuando una aplicación de software (400) busca un objeto que está fuera de validez 405 en una caché local, es retirado y, por lo tanto, no es devuelto a la aplicación solicitante. Esto pone en marcha la interrogación de un CDS del nivel intermedio (el seleccionado por la función de equilibrado de carga, no mostrado, tal como se explica en la figura 3). Si el objeto se encuentra presente en la caché central (445) y no está fuera de validez, es enviado por el gestionador de datos (450), tal como se ha explicado anteriormente, a la caché local de la aplicación solicitante en la que se almacena (425) y de la que se puede suministrar (435) a la aplicación solicitante. Esto termina el proceso de sustitución de un objeto fuera de validez en una caché local. Si una copia nueva del objeto solicitado es fácilmente disponible en una caché central, ello resulta muy probablemente del hecho de que otra aplicación, utilizando otra caché local, ha necesitado ya el mismo objeto que ha sido previamente reconstruido en CDS.

No obstante, si no es este el caso, es decir, si el objeto solicitado está también fuera de validez en la caché central (455), se debe reconstruir. Esto se lleva a cabo, tal como ya se ha explicado en la figura 3, a partir de la base de datos (465) por el gestionador de datos que requiere también la invalidación (470) del objeto que está fuera de validez con intermedio de una función de gestionador de invalidación (460). Por lo tanto, el objeto que está fuera de validez es sustituida eventualmente en la caché central por otro de nueva construcción (475) y se duplica también en todos los CDS del nivel intermedio, tal como se ha explicado en la figura 3. Es importante observar en este caso que si el CDS del que se ha requerido un objeto fuera de validez (455) no es el CDS principal, sino un CDS de refuerzo, como resultado de la elección llevada a cabo por la función de equilibrado de carga mencionado en la figura 3, la reconstrucción no es llevada a cabo realmente por el gestionador de datos de refuerzo. En vez de ello, la petición de

invalidación de objeto es enviada (480) al gestionador de invalidación del CDS principal, de manera que es el CDS principal el que lleva a cabo la reconstrucción y replica el objeto reconstruido en todos los refuerzos a efectos de que todos los CDS del nivel intermedio están eventualmente actualizados incluyendo aquél que ha recibido la petición original.

Dado que la reconstrucción puede ser un proceso largo, el objeto solicitado originalmente (455) aunque esté fuera de validez, es suministrado a pesar de ello a la caché local peticionaria (485), de manera que la aplicación peticionaria (400) no quede bloqueada. Antes del suministro, no obstante, el TTL del objeto es dispuesto en un valor muy bajo. Por lo tanto, el objeto es todavía utilizable para la petición o peticiones actuales mientras se está efectuando la reconstrucción, tal como se ha explicado anteriormente. Las otras peticiones, recibidas después de que ha expirado el reducido TTL, utilizarán, por lo tanto, el objeto que se ha reconstruido de nuevo. Esta forma de operar es compatible con el tipo de aplicación considerada por la aplicación, en la que las tarifas aéreas son actualizadas regularmente, no obstante, con tiempos de actualización flexibles, puesto que requiere un tiempo significativo el desplegar un nuevo conjunto de tarifas aéreas mediante una red grande.

La **figura 5** describe la invalidación de objetos en cachés cuando se modifican elementos de datos en las bases de datos (500). Esto ocurre cuando la base de datos es actualizada por sus usuarios (505), por los que están en cargo de actualizar y gestionar su contenido. El servidor de la base de datos es capaz de detectar qué elementos de datos, que fueron utilizados para construir objetos, han cambiado. El conjunto de objetos afectados son determinados a efectos de invalidarlos y reconstruirlos, posiblemente en las cachés centrales. Con este objetivo, la base de datos envía instrucciones (510) de invalidación de objeto a todos los CDS, es decir, CDS principal y CDS de refuerzo o refuerzos, en los que todos los objetos involucrados son invalidados (515, 525).

En el CDS principal (520), el gestionador de invalidación (530) tiene que decidir, en primer lugar, si se tiene que anular un objeto invalidado o tiene que ser reconstruido. Ciertamente, un objeto todavía presente en la caché central puede no merecer ser reconstruido. Por ejemplo, si no ha sido utilizado durante un largo tiempo o, es utilizado raramente, o es invalidado frecuentemente, el gestionador de invalidación puede decidir simplemente eliminarlo a efectos de no ocupar innecesariamente memoria caché y ahorrar potencia de proceso y ancho de banda de comunicación necesarios para la reconstrucción. La decisión es llevada a cabo de acuerdo con normas predefinidas. El funcionamiento del gestionador de invalidación es definido y dispuesto por los administradores del CDS.

Entonces, si se hace la decisión de suprimir el objeto, el espacio de memoria caché correspondiente de la caché queda liberado. No obstante, si el gestionador de invalidación decide que el objeto debe ser reconstruido, ello se lleva a cabo a partir de la base de datos, de manera similar a la ya explicada en la figura 3. Una vez efectuada la reconstrucción (525), el objeto es replicado (540) en todos los CDS, de manera que son sincronizados y una copia nueva queda disponible de todas las cachés centrales (545).

Tanto si se lleva a cabo la supresión o reconstrucción del objeto, el gestionador de invalidación del CDS principal transmite invalidaciones de objeto a todos los satélites (550) para impedir que las aplicaciones de software (555) de los servidores del primer nivel utilicen objetos fuera de validez y obsoletos (560). Cuando la aplicación de software requiere nuevamente un objeto invalidado, que por lo tanto ya no se encuentra disponible en la caché local, tendrá lugar la búsqueda del objeto del CDS, y posiblemente su reconstrucción, tal como ya se ha explicado en la figura 3.

Asimismo, después de completar la eliminación o reconstrucción del objeto, las invalidaciones son notificadas (570) desde el CDS principal al CDS o a los CDS de refuerzo. Esto es realizado para llevar a cabo la gestión de una tabla de avance utilizada por el gestionador de invalidación, tal como se explica en la figura siguiente.

La **figura 6** explica además la reconstrucción de un objeto invalidado.

Cuando esto tiene lugar, tal como se ha explicado en la figura 5, existe una tabla de avance (600) en los CDS, que es actualizada al recibir una petición de invalidación de un objeto. La operación que se está realizando, es decir, la reconstrucción (605) de un objeto invalidado, es archivada en la tabla (610) del CDS principal, y enviado también a las tablas de refuerzo en avance (625). Asimismo, el TTL o fecha de expiración del objeto actual invalidado es cambiada a un valor bajo, de manera que es todavía utilizable mientras el nuevo objeto está siendo reconstruido, tal como se ha explicado anteriormente.

La terminación normal de esta operación es que el objeto es normalmente reconstruido y pasa todas las verificaciones y comprobaciones (640). En este caso, es enviado (615) a los CDS de refuerzo, tal como ya se ha explicado, de manera que resulta disponible en todas las cachés centrales. Al mismo tiempo, el registro de la operación en avance es borrado de la tabla principal en avance, y también de los CDS de refuerzo (620), que reciben el objeto reconstruido (615).

Estas operaciones pueden no terminar posiblemente de modo normal, sino afectando el buen funcionamiento del CDS a largo plazo.

Una primera situación a considerar es cuando un objeto en el CDS principal no puede ser reconstruido completamente, por ejemplo, porque los elementos de datos necesarios no fueron transferidos de manera apropiada desde la base de datos (630). La operación de reconstrucción debe ser intentada nuevamente, cuyo inicio ha sido registrado en la tabla de avance (500). La operación pendiente es detectada entonces por un proceso de barrido (635) que controla e inspecciona periódicamente la tabla de avance. Existe, evidentemente, un límite para el número de nuevos intentos realizados por el gestionador de invalidación para la operación pendiente registrada en la tabla de avance. En condiciones normales, uno de los nuevos intentos tiene éxito y el funcionamiento se reanuda normalmente, tal como se ha explicado en lo anterior.

Una segunda situación a considerar es cuando el objeto que se acaba de reconstruir (610) no es transferido normalmente a ninguno de los CDS de refuerzo. Esto es detectado de manera similar que en el CDS principal, puesto que los CDS de refuerzo tienen también todos una tabla de avance (620). Por lo tanto, el proceso de barrido de refuerzo detecta la operación pendiente que ha sido registrada como resultado de la invalidación de objeto enviada desde la base de datos. Los CDS de refuerzo que esperan que el CDS principal reconstruya el objeto invalidado no llevan a cabo normalmente ninguna reconstrucción de objeto. No obstante, esto puede ser puesto en marcha excepcionalmente por el gestionador de invalidación de refuerzo (650) cuando el gestionador de barrido informa de un problema. Cuando se encuentra esta situación, el gestionador de invalidación de refuerzo, eventualmente, reconstruye (655) el objeto invalidado para su propia caché, comportándose temporalmente como CDS principal. Esto requiere la transferencia (al CDS de refuerzo) de los elementos de datos necesarios desde la base de datos. Después de ello, el elemento pendiente de la tabla de avance puede ser borrado y el funcionamiento puede reanudarse normalmente para los CDS de refuerzo afectados.

La **figura 7** describe brevemente una característica opcional de la invención.

Una caché tiene necesariamente un tamaño finito dado que está construido a base de la memoria activa disponible del servidor central en el que reside. Por lo tanto, puede ocurrir que la caché pase a estar llena: no queda suficiente memoria para construir un nuevo objeto solicitado. Una práctica estándar con las cachés consiste entonces en eliminar el objeto utilizado menos recientemente (LRU) para dejar sitio para el nuevo objeto. También se pueden aplicar otros algoritmos de sustitución. Para cualquier algoritmo utilizado, el objeto seleccionado para ser eliminado, que no ha sido utilizado recientemente, puede ser todavía un objeto válido, es decir, no está fuera del periodo de utilización. Tal como se ha indicado, los objetos, según la invención, son objetos complejos que requieren una potencia de proceso significativa de la unidad de proceso central (CPU) del servidor para ser construidos, y requieren también múltiples accesos a las bases de datos del historial (700). Por lo tanto, la reconstrucción de un objeto es costosa y se debe evitar descartarlo si el objeto no está fuera de validez cuando se retira de la caché.

Cuando una caché central se encuentra llena (760), tanto en un CDS principal (720) como en cualquiera de los CDS de refuerzo (730), y por ejemplo, se ha seleccionado el objeto LRU (770) a eliminar, éste último puede ser almacenado (775) en una memoria secundaria (740, 750), fácilmente accesible desde el servidor si no está fuera de plazo (792), en vez de ser descartado (780). Si el objeto está fuera de plazo (794), no obstante, debe ser descartado. La memoria secundaria es, en general, un espacio de memoria destinado de un disco duro acoplado al servidor. Por lo tanto, la reconstrucción de objetos que no están fuera de validez puede ser evitada. Para ello, los CDS efectúan el seguimiento de los objetos almacenados en las memorias secundarias. Cuando una aplicación necesita uno de ellos nuevamente, es recuperado de la memoria secundaria correspondiente en vez de ser reconstruido. El mismo mecanismo es aplicable, es decir, el objeto LRU es retirado de la caché para dejar sitio al objeto solicitado, que es buscado, en primer lugar, en memoria secundaria antes de ser reconstruido o si la búsqueda no tiene éxito o si el objeto ha pasado a estar fuera de validez desde que fue almacenado en la memoria secundaria.

REIVINDICACIONES

1. Método para mantener la coherencia de contenidos caché en una arquitectura de software de niveles múltiples, que comprende un primer nivel (220) dotado, como mínimo, de un servidor/cliente, operando cada uno de dichos, como mínimo, un servidor/cliente, una caché local (230), incluyendo dicha arquitectura un nivel intermedio (210) que comprende, como mínimo, un servidor central (212), operando cada uno de dichos, como mínimo, un servidor central, una caché central (216), interconectándose cada uno de dichos, como mínimo, un servidor central con, como mínimo, una base de datos (204) a través de, como mínimo, un servidor de base de datos (202), conteniendo dicho, como mínimo, una base de datos, elementos de datos (360), siendo recuperados dichos datos con peticiones emitidas desde cada uno de dichos, como mínimo, un servidor central a dicho, como mínimo, un servidor de la base de datos, cuyo método comprende las siguientes etapas de:

- construir (350) objetos en dicho, como mínimo, un servidor central desde, como mínimo, uno de dichos elementos de datos procedentes de, como mínimo, una base de datos, incluyendo en dicha etapa de construcción las etapas adicionales de: validar dichos objetos; atribuir un tiempo de validez TTL a dichos objetos y almacenar dichos objetos en dichas cachés centrales (340);

- enviar (330, 370) dichos objetos desde dichas cachés centrales a cualquiera de dichos servidores de cliente que solicitan dichos objetos, incluyendo dicha etapa de envío la etapa adicional de: almacenar dichos objetos en dichas cachés locales;

- suministrar (335, 375) dichos objetos a las aplicaciones de software utilizadas en dichos servidores de clientes y que solicitan dichos objetos;

y que se caracteriza porque los objetos almacenados en dichas cachés centrales y en dichas cachés locales son invalidados tan pronto como, como mínimo, uno de dichos elementos de datos utilizados para construir dichos objetos (500) ha sido modificado en dicha base de datos, comprendiendo el método además las etapas de:

- determinar los objetos afectados;

- emitir solicitudes de invalidación (510) a todos los gestionadores invalidación de dichos servidores centrales;

- invalidar los objetos correspondientes (515) en dichas cachés centrales;

- propagar dichas instrucciones de invalidación (550) a todas las mencionadas cachés locales;

- invalidar los objetos correspondientes (560) en dichas cachés locales, incluyendo dicha etapa de invalidación la etapa adicional de borrado de dichos objetos.

2. Método, según la reivindicación 1, en el que las peticiones de dichos servidores de clientes son equilibradas en cuanto a carga (385) con respecto a la totalidad de dichos servidores centrales y en el que uno de dichos servidores centrales es seleccionado por cada petición.

3. Método, según las reivindicaciones 1 ó 2, en el que dicha etapa de construcción comprende la etapa adicional de: replicar (380) un objeto (355) de nueva construcción en todos los demás servidores centrales a partir de dicho servidor central seleccionado.

4. Método, según las reivindicaciones 1 ó 2, en el que un objeto es solicitado (390) de dicha caché central seleccionada (340) siempre que falte (320, 350) o esté fuera del plazo de validez (405) en dicha caché local.

5. Método, según la reivindicaciones 1 ó 2, en el que dicha etapa de construcción (350) es puesta en marcha en dicho servidor central seleccionado siempre que el objeto solicitado falte (355) en dicha caché central seleccionada.

6. Método, según cualquiera de las reivindicaciones 1 a 5, que comprende una serie de servidores centrales y en el que uno de dichos servidores centrales está diseñado como servidor central principal (340) y todos los demás son servidores centrales de refuerzo.

7. Método, según la reivindicación 6, en el que dicha etapa de construcción es puesta en marcha en dicho servidor central principal (480) desde un gestionador de invalidación (460) siempre que dicho objeto solicitado se encuentre fuera de plazo de validez (455) en dicho servidor central seleccionado.

8. Método, según la reivindicación 7, en el que el TTL de dicho objeto fuera de validez (455) recibe un valor bajo antes de ser enviado (485) al servidor del cliente peticionario.

9. Método, según cualquiera de las reivindicaciones anteriores, en el que la etapa de invalidar objetos en dichas cachés centrales comprende la etapa adicional de borrar dichos objetos.

10. Método, según la reivindicación 6, en el que la etapa de invalidar objetos comprende la etapa de reconstruir dichos objetos (535) en dicho servidor central principal (520), comprendiendo la etapa adicional de replicar (540) dichos objetos reconstruidos en la totalidad de dichos servidores centrales de refuerzo.

11. Método, según la reivindicación anterior, en el que cada uno de dichos servidores centrales comprende una tabla de avance (600) de la reconstrucción de objetos y un proceso de barrido (635) para controlar dicha reconstrucción de objetos a partir de la tabla de avance, comprendiendo las etapas de:

- registrar el principio y final de las reconstrucciones de objetos en dicha tabla de avance (610);

- inspeccionar dichas tablas de avance (635) para detectar la terminación de dichas reconstrucciones de objetos, comprendiendo la etapa adicional de: borrar las entradas de la tabla de las reconstrucciones de objetos que han sido terminadas satisfactoriamente;

- detectar fallos en la reconstrucción de objetos en avance;

- intentar reconstrucciones de objetos en fallo.

12. Método, según la reivindicación anterior, en el que dichas reconstrucciones de objetos son llevadas a cabo en dicho servidor central principal y en el que dicha etapa de registro incluye la etapa adicional de: enviar inicios de reconstrucción de objetos (625) a la totalidad de dichas tablas de avance de los servidores centrales de refuerzo.

13. Método, según la reivindicación 12, en el que la terminación de la reconstrucción de objetos es registrada en dichas tablas de avance de los servidores centrales de refuerzo al recibir (615) los objetos enviados desde el servidor central principal.

14. Método, según la reivindicación 11, en el que dicha etapa de detección y dicha etapa de nuevo intento son llevadas a cabo en uno de dichos servidores centrales de refuerzo.

15. Método, según la reivindicación 1, en el que dicha etapa de reconstrucción incluye las etapas preliminares de: comprobar si dicha caché central está completa o no; en caso negativo proceder inmediatamente a dicha etapa de construcción; si está completa (760), seleccionar el objeto Utilizado Menos Recientemente (770) en dicha caché central; y

comprobar si dicho objeto Utilizado Menos Recientemente está fuera de validez o no; en caso negativo (792), almacenar dicho objeto Utilizado Menos Recientemente en una segunda memoria (775); y si está fuera de validez (794), descartar (780) dicho objeto Utilizado Menos Recientemente, procediendo en ambos casos a dicha etapa de construcción.

16. Método, según la reivindicación 10, en el que se efectúa la búsqueda en una memoria secundaria (720) de un objeto faltante, antes de proceder a dicha etapa de reconstrucción.

17. Sistema para mantener la coherencia de contenidos caché en una arquitectura de servidores de niveles múltiples, comprendiendo, dicho sistema, medios adaptados para llevar a cabo cada etapa del método según cualquiera de las reivindicaciones 1 a 16.

18. Producto de programa de ordenador almacenado en un medio de almacenamiento legible por ordenador, que comprende medios de código legibles por ordenador para provocar, como mínimo, que un ordenador lleve a cabo el método de mantener la coherencia del contenido caché en un sistema de software de niveles múltiples, de acuerdo con cualquiera de las reivindicaciones 1 a 16.

Técnica anterior

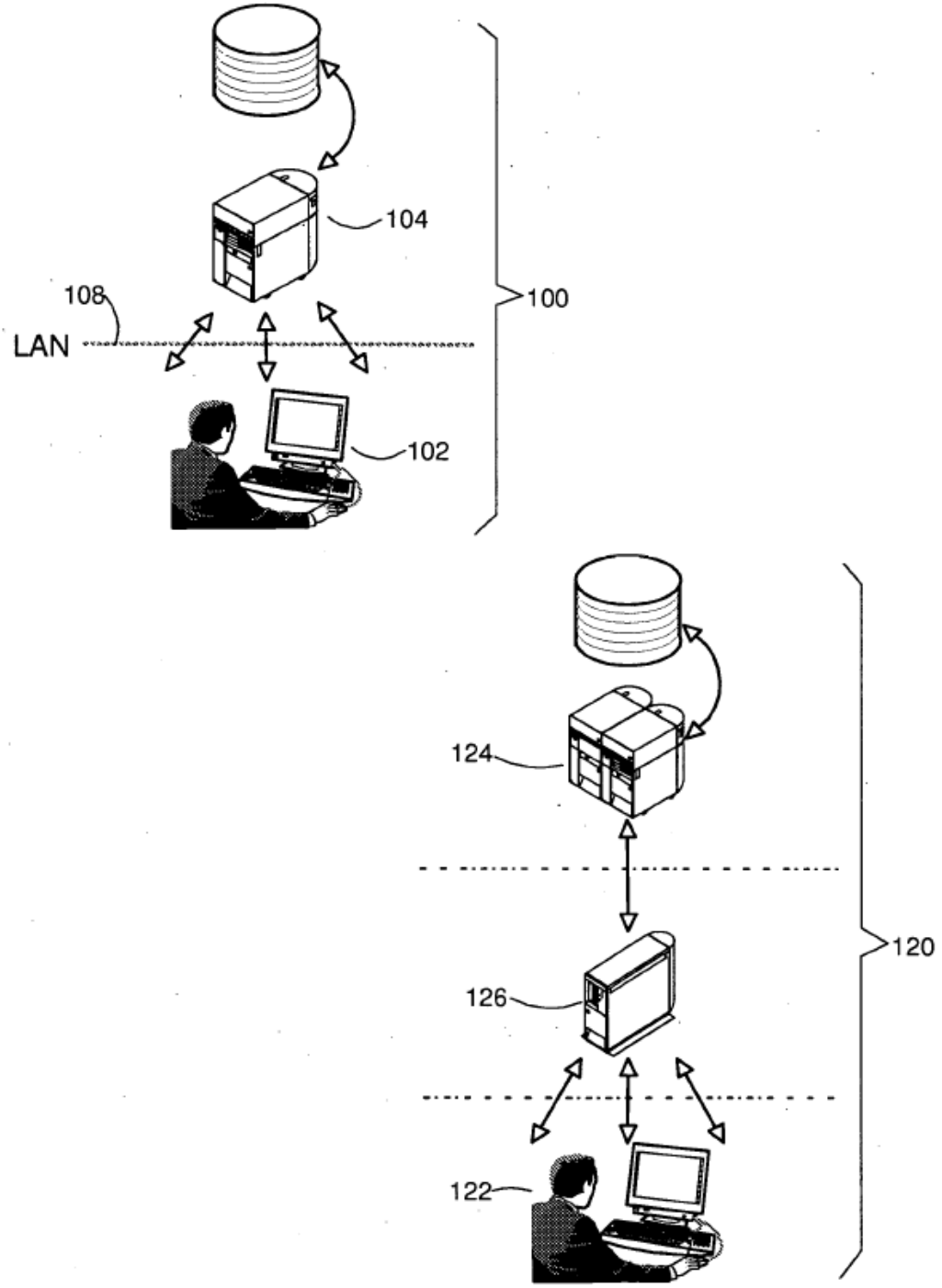

Figura 1

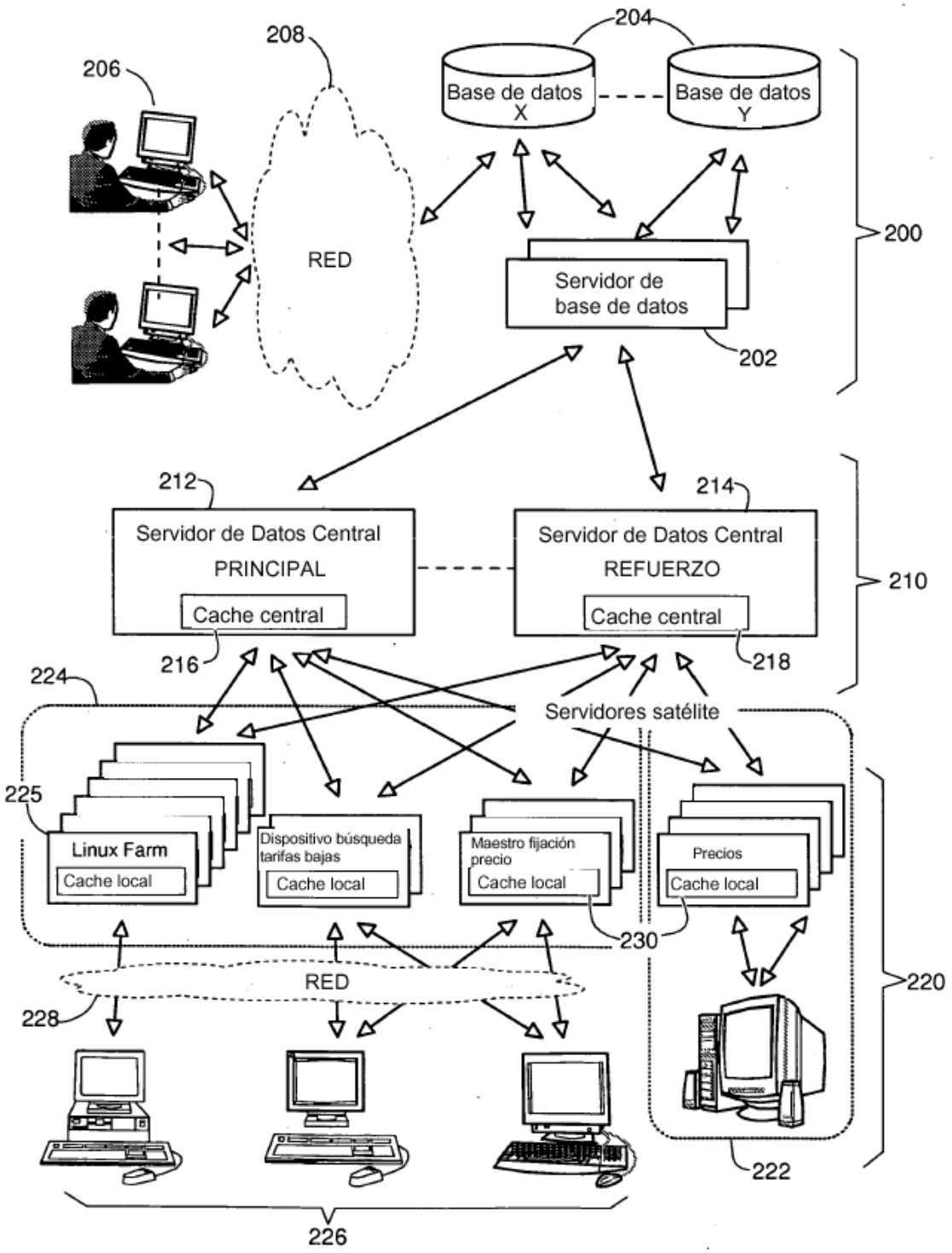
208
204
206
Base de datos X
Base de datos Y
RED
200
Servidor de base de datos
202
212
214
Servidor de Datos Central PRINCIPAL
Cache central
Servidor de Datos Central REFUERZO
Cache central
210
216
218
224
Servidores satélite
225
Linux Farm
Cache local
Dispositivo búsqueda tarifas bajas
Cache local
Maestro fijación precio
Cache local
Precios
Cache local
230
220
RED
228
222
226

Figura 2

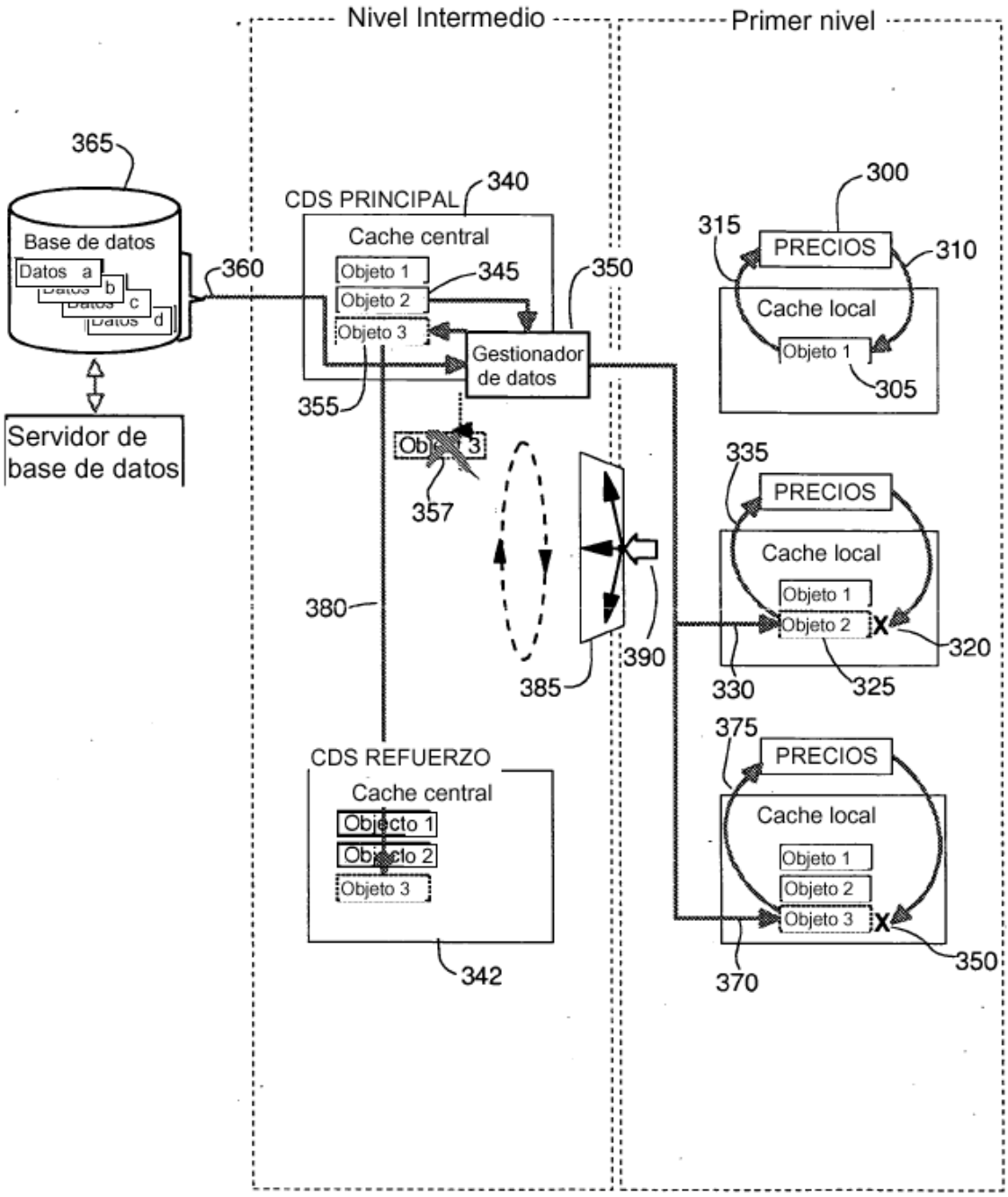
Nivel Intermedio
Primer nivel
365
Base de datos
Datos a
b
c
d
360
Servidor de base de datos
CDS PRINCIPAL
340
Cache central
Objeto 1
Objeto 2
Objeto 3
345
350
Gestionador de datos
355
357
380
385
390
CDS REFUERZO
Cache central
Objecto 1
Objeto 2
Objeto 3
342
300
315
PRECIOS
310
Cache local
Objeto 1
305
335
PRECIOS
Cache local
Objeto 1
Objeto 2
320
330
325
375
PRECIOS
Cache local
Objeto 1
Objeto 2
Objeto 3
350
370

Figura 3

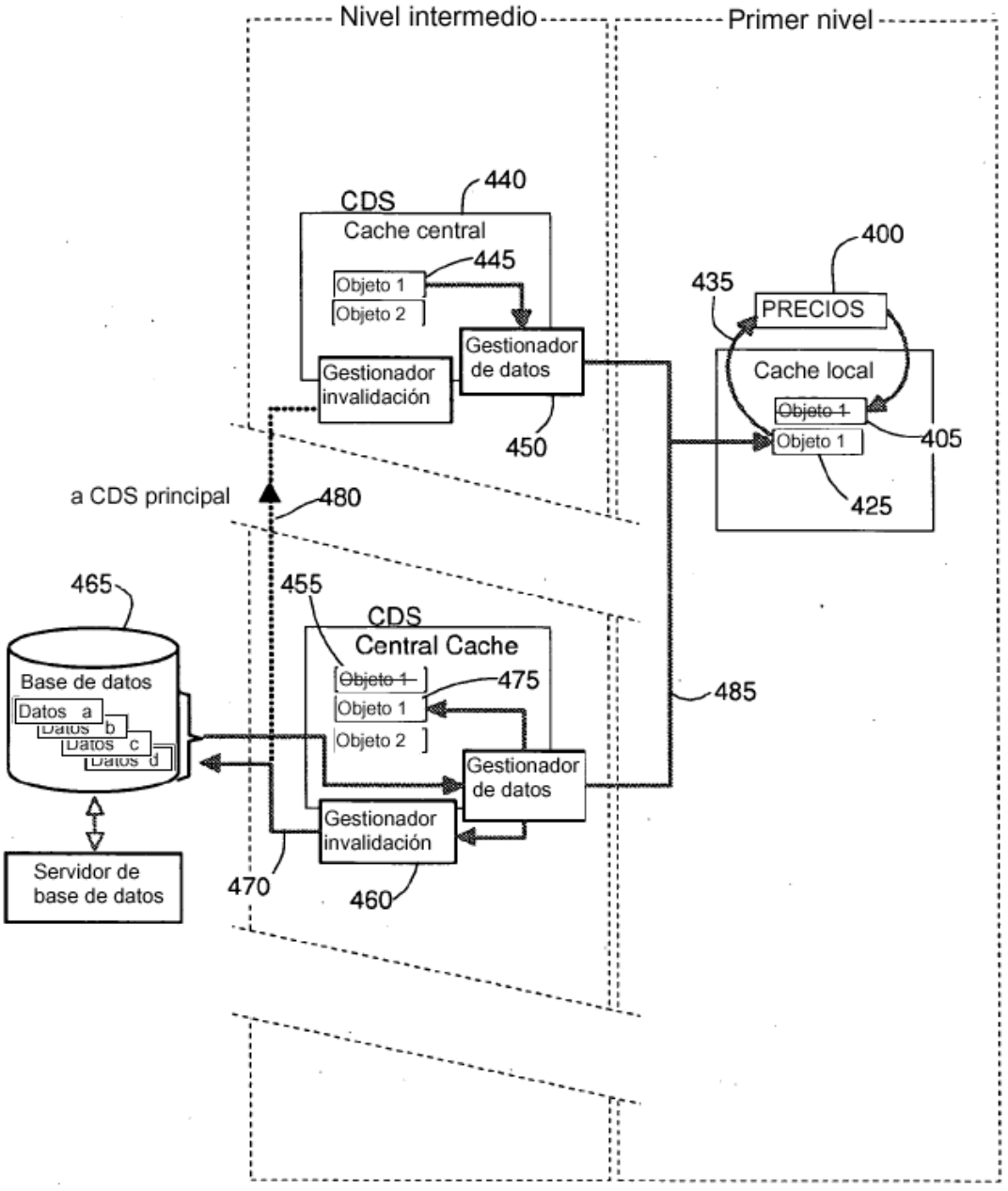
Nivel intermedio
Primer nivel
CDS
Cache central
440
445
Objeto 1
Objeto 2
Gestionador de datos
Gestionador invalidación
450
400
435
PRECIOS
Cache local
Objeto 1
Objeto 1
405
425
a CDS principal
480
465
455
CDS
Central Cache
Objeto 1
Objeto 1
475
Objeto 2
485
Base de datos
Datos a
Datos b
Datos c
Datos d
Gestionador de datos
Gestionador invalidación
Servidor de base de datos
470
460

Figura 4

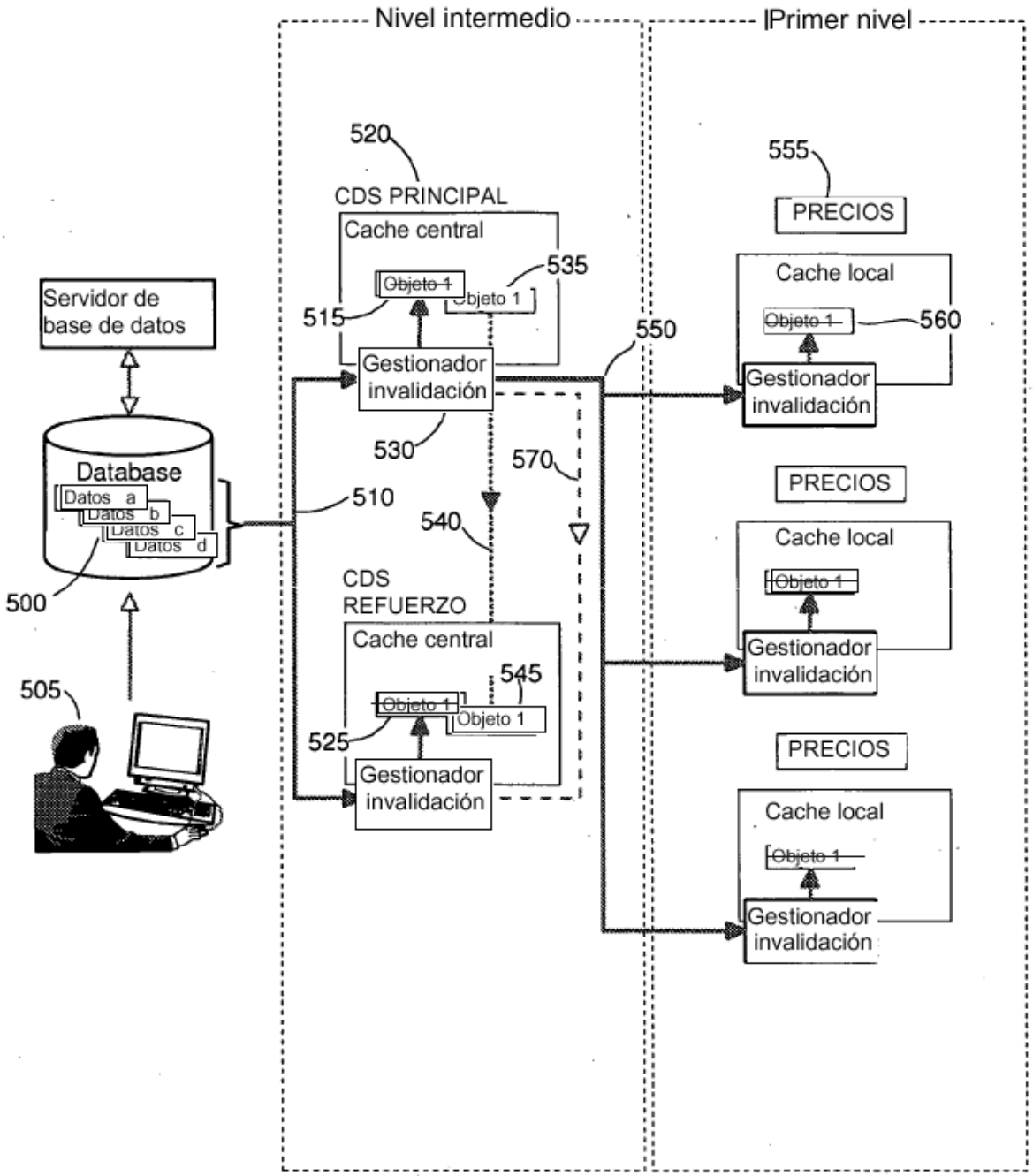
Nivel intermedio
Primer nivel
520
CDS PRINCIPAL
Cache central
Objeto 1
Objeto 1
535
515
Gestionador invalidación
530
550
570
Servidor de base de datos
Database
Datos a
Datos b
Datos c
Datos d
510
500
540
CDS REFUERZO
Cache central
545
Objeto 1
Objeto 1
525
Gestionador invalidación
505
555
PRECIOS
Cache local
Objeto 1
560
Gestionador invalidación
PRECIOS
Cache local
Objeto 1
Gestionador invalidación
PRECIOS
Cache local
Objeto 1
Gestionador invalidación

Figura 5

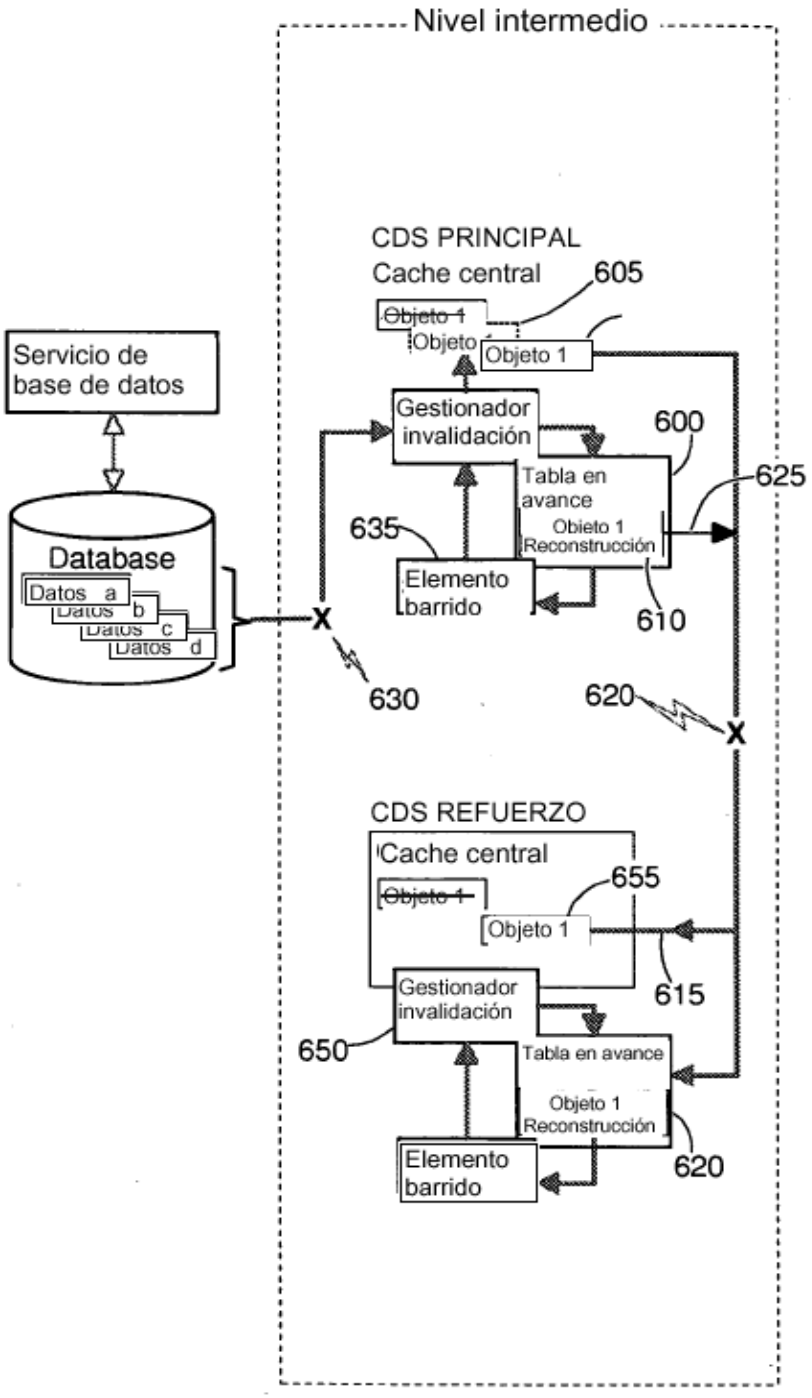
Nivel intermedio
CDS PRINCIPAL
Cache central
605
Objeto 1
Objeto
Objeto 1
Servicio de base de datos
Gestionador invalidación
600
625
Tabla en avance
Objeto 1 Reconstrucción
635
Elemento barrido
610
Database
Datos a
Datos b
Datos c
Datos d
X
630
620
X
CDS REFUERZO
Cache central
655
Objeto 1
Objeto 1
615
Gestionador invalidación
650
Tabla en avance
Objeto 1 Reconstrucción
Elemento barrido
620

Figura 6

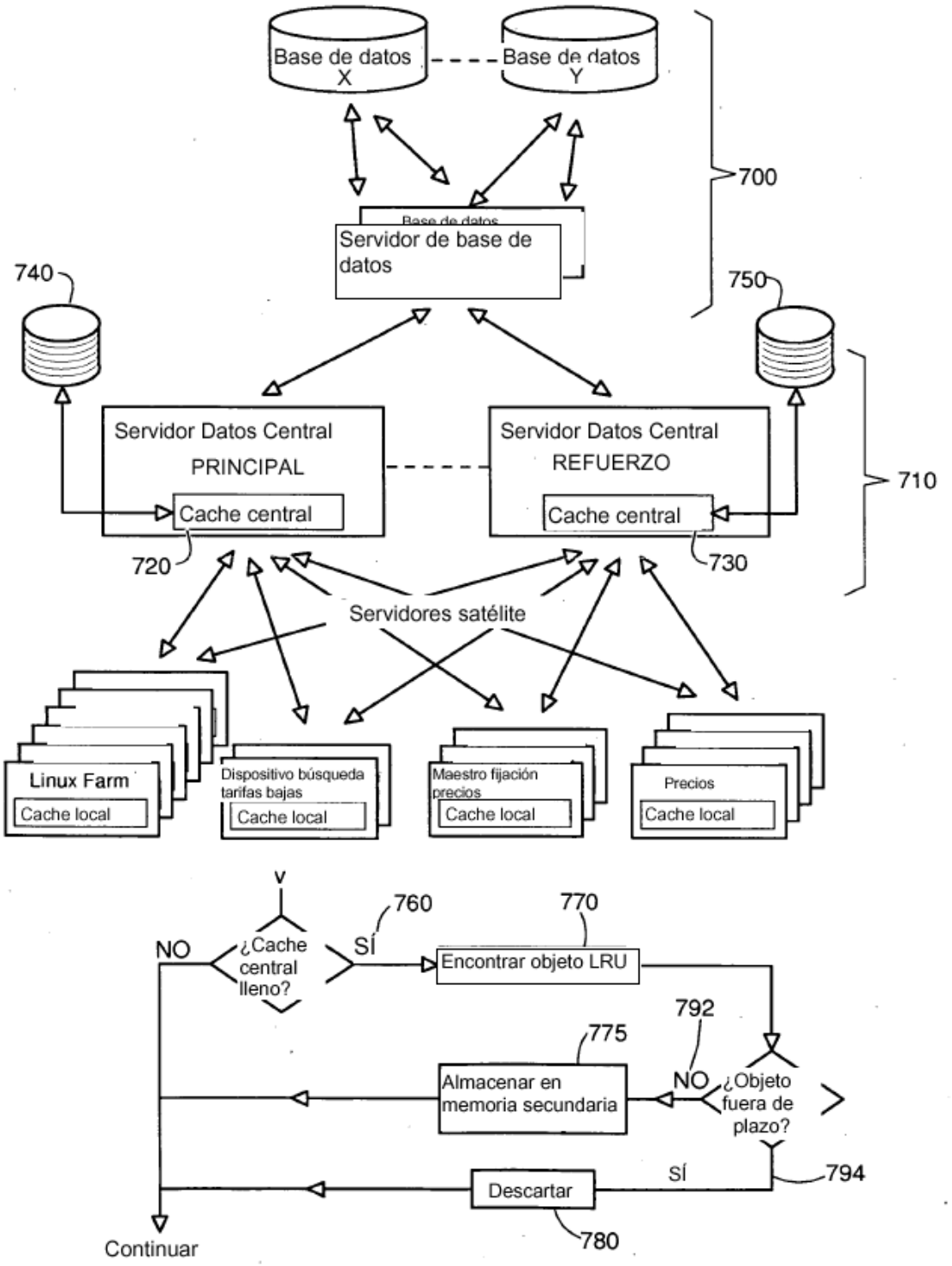
Base de datos X
Base de datos Y
700
Servidor de base de datos
740
750
Servidor Datos Central PRINCIPAL
Cache central
Servidor Datos Central REFUERZO
Cache central
710
720
730
Servidores satélite
Linux Farm
Cache local
Dispositivo búsqueda tarifas bajas
Cache local
Maestro fijación precios
Cache local
Precios
Cache local
760
770
NO
¿Cache central lleno?
SÍ
Encontrar objeto LRU
792
775
Almacenar en memoria secundaria
NO
¿Objeto fuera de plazo?
SÍ
794
Descartar
780
Continuar

Figura 7