

19



OFICINA ESPAÑOLA DE
PATENTES Y MARCAS

ESPAÑA



11 Número de publicación: **2 909 440**

51 Int. Cl.:

G06F 16/903

(2009.01)

12

TRADUCCIÓN DE PATENTE EUROPEA

T3

86 Fecha de presentación y número de la solicitud internacional: **02.12.2014 PCT/US2014/068191**

87 Fecha y número de publicación internacional: **11.06.2015 WO15084864**

96 Fecha de presentación y número de la solicitud europea: **02.12.2014 E 14867565 (5)**

97 Fecha y número de publicación de la concesión europea: **23.02.2022 EP 3077928**

54 Título: **Sistemas y métodos de modelado de redes de objetos**

30 Prioridad:

02.12.2013 US 201361910799 P
01.12.2014 US 201414557248

45 Fecha de publicación y mención en BOPI de la
traducción de la patente:
06.05.2022

73 Titular/es:

RAKUTEN GROUP, INC. (100.0%)
1-14-1 Tamagawa, Setagaya-ku
Tokyo 158-0094, JP

72 Inventor/es:

LYSNE, STIAN, K.J.;
PELLEGRINI, MICHAEL y
LAUKLI, BJORN, A.

74 Agente/Representante:

VIDAL GONZÁLEZ, Maria Ester

ES 2 909 440 T3

Aviso: En el plazo de nueve meses a contar desde la fecha de publicación en el Boletín Europeo de Patentes, de la mención de concesión de la patente europea, cualquier persona podrá oponerse ante la Oficina Europea de Patentes a la patente concedida. La oposición deberá formularse por escrito y estar motivada; sólo se considerará como formulada una vez que se haya realizado el pago de la tasa de oposición (art. 99.1 del Convenio sobre Concesión de Patentes Europeas).

DESCRIPCIÓN

Sistemas y métodos de modelado de redes de objetos

5 Referencia cruzada a las solicitudes relacionadas y reivindicación de prioridad

Esta solicitud reivindica prioridad para Solicitud de patente provisional de Estados Unidos con Núm. de serie 61/910,799, presentada el 2 de diciembre de 2013, compartiendo el mismo título, cuyo contenido completo se incorpora por referencia en la presente descripción.

10 Campo

El campo técnico se relaciona generalmente con la recuperación de información basada en asociaciones y, más particularmente, con sistemas y métodos para modelar asociaciones entre objetos dentro de una red de objetos.

15 Antecedentes

Los sistemas de información que descubren y exploran asociaciones entre entidades descritas por datos subyacentes abundan en las economías basadas en la información de hoy. Ejemplos de estos sistemas de información incluyen almacenes de datos, sistemas de inteligencia comercial, motores de búsqueda y similares. Muchos de estos sistemas utilizan sistemas de gestión de bases de datos comerciales, como ORACLE 12c disponible de Oracle Corporation de Redwood City, California. Otros sistemas, como el motor de búsqueda GOOGLE disponible de Google Inc. de Mountain View, California, utilizan sistemas de bases de datos patentados, como BigTable y Spanner.

25 Sin embargo, los sistemas de información convencionales, incluidos los sistemas de bases de datos convencionales, se han quedado atrás de las capacidades mejoradas de hardware y software actuales. Dicho de otra manera, los sistemas de información convencionales no están diseñados para aprovechar la potencia de procesamiento mejorada y el espacio de memoria direccionable expandido actualmente disponible o, más aún, la potencia de procesamiento mejorada y el espacio de memoria direccionable expandido en el horizonte.

30 Por ejemplo, ahora existen varias CPU que son capaces de implementar la versión de 64 bits del conjunto de instrucciones x86 (también conocido como x64). Como apreciarán los expertos en la técnica, estos sistemas informáticos x64 incluyen conjuntos mucho más grandes de memoria direccionable en comparación con los sistemas informáticos convencionales. Además, se anticipa que las extensiones de 512 bits a las instrucciones SIMD (datos múltiples de instrucción única) de extensiones vectoriales avanzadas de 256 bits para la arquitectura del conjunto de instrucciones x86 estarán disponibles en los próximos años. Un ejemplo de dichas extensiones de 512 bits que se espera con entusiasmo son las extensiones AVX-512 propuestas por Intel en julio de 2013. Sin embargo, los sistemas de información convencionales están mal equipados para aprovechar las capacidades de procesamiento en paralelo que estarán disponibles, por ejemplo, las extensiones AVX-512 y los procesadores configurados para ejecutar lo mismo. Además, se anticipa que las GPGPU (unidades de procesamiento de gráficos de uso general) representarán opciones de hardware atractivas para realizar un procesamiento altamente eficiente en el futuro debido a su arquitectura, que se presta al procesamiento paralelo masivo. Sin embargo, los sistemas de información convencionales no están diseñados para aprovechar el poder de cómputo que está disponible actualmente o que estará disponible en poco tiempo.

45 En consecuencia, lo que se necesita son sistemas informáticos y métodos de modelado de redes de objetos que den cuenta de las espectaculares mejoras en el hardware y el software informático disponibles en la actualidad y en el horizonte próximo. El documento EP 2 130 953 A2 se refiere a un sistema de gestión de imágenes para buscar datos de imágenes. El documento WO 2013/044295 A1 se relaciona con un sistema para unir código hash a una parte de una imagen. Identificar objetos similares a un objeto de sondeo en el documento US 2010/179933 implica generar vectores externos a partir de los objetos, generar vectores internos a partir de los vectores externos y generar huellas digitales a partir de los vectores internos.

50 Resumen

De acuerdo con aspectos y realizaciones de la presente invención, se proporciona el sistema informático de la reivindicación 1. Otras realizaciones son objeto de las reivindicaciones dependientes 2 a la 6.

60 Debe entenderse que la siguiente descripción detallada proporciona meramente ejemplos ilustrativos de varios aspectos y realizaciones, y pretende proporcionar una visión general o marco para comprender la naturaleza y el carácter de los aspectos y realizaciones reivindicadas. Cualquier realización descrita en la presente descripción puede combinarse con cualquier otra realización. Referencias a "una realización", "un ejemplo", "algunas realizaciones", "algunos ejemplos", "una realización alternativa", "varias realizaciones", "una realización", "al menos una realización", "otra realización", "otras realizaciones", "esta y otras realizaciones" o similares no son necesariamente excluyentes entre sí y pretenden indicar que una característica, estructura o característica particular descrita en relación con la

realización puede incluirse en al menos una realización. Las apariciones de dichos términos en la presente descripción no se refieren necesariamente a la misma realización.

Breve descripción de los dibujos

5 Varios aspectos de al menos una realización se discuten a continuación con referencia a las figuras adjuntas, que no pretenden estar dibujadas a escala. Las figuras se incluyen para proporcionar una ilustración y una mayor comprensión de los diversos aspectos y realizaciones, y se incorporan y constituyen parte de esta descripción, pero no pretenden ser una definición de los límites de ninguna realización en particular. Los dibujos, junto con el resto de la descripción,
10 sirven para explicar los principios y operaciones de los aspectos y realizaciones descritas y reivindicadas. En las figuras, cada componente idéntico o casi idéntico que se ilustra en varias figuras se representa por un número similar. Para mayor claridad, no todos los componentes pueden etiquetarse en todas las figuras. En las figuras:

15 La Figura 1 es un diagrama de bloques de un sistema informático distribuido;
La Figura 2 es un diagrama de bloques que muestra un sistema para modelar redes de objetos;
La Figura 3 es un diagrama de relación de entidades que define varias relaciones entre objetos en una red de objetos;
La Figura 4 es un diagrama de relación de entidades que define varias relaciones entre objetos en una red de objetos;
20 La Figura 5 es un diagrama de relación de entidades de una red de objetos de ejemplo;
Las Figuras 6 a la 8 son una secuencia de diagramas de relación de entidades que ilustran un proceso para unir un objeto a un centroide;
La Figura 9 es un diagrama de relación de entidades de una red de objetos de ejemplo;
La Figura 10 es un diagrama de relación de entidades de una red de objetos de ejemplo;
25 La Figura 11 es un diagrama de relación de entidades de una red de objetos de ejemplo;
La Figura 12 es un diagrama de estructura de datos que ilustra un vector interno;
La Figura 13 es un diagrama de secuencia que ilustra un proceso de codificación de vectores;
La Figura 14 es un gráfico que ilustra el cálculo de similitud de coseno;
La Figura 15 es un diagrama de estructura de datos que ilustra un conjunto de índices de permutación;
30 La Figura 16 es un diagrama de secuencia que ilustra un proceso de generación de huellas digitales;
La Figura 17 es un diagrama de secuencia que ilustra un proceso de generación de LSH;
La Figura 18 es un diagrama de secuencia que ilustra un proceso de reducción de datos;
La Figura 19 es un diagrama de secuencia que ilustra un proceso de consulta;
La Figura 20 es un diagrama de Venn modificado que ilustra un proceso de recuperación;
35 La Figura 21 es un diagrama de bloques que destaca los componentes implicados en un proceso de recuperación de similitudes;
La Figura 22 es un diagrama de secuencia que ilustra un proceso de recuperación de similitudes;
La Figura 23 es un diagrama de entidad de datos que ilustra asignadores de vectores;
La Figura 24 es un diagrama de bloques que destaca los componentes implicados en un proceso de recuperación dimensional;
40 La Figura 25 es un diagrama de bloques que destaca los componentes implicados en un proceso de recuperación de fusión;
La Figura 26 es un diagrama de Venn modificado que ilustra un proceso de precisión;
La Figura 27 es un diagrama de Venn modificado que ilustra un proceso de antisolapamiento;
45 La Figura 28 es un diagrama de Venn de precisión de consulta en vista de configuraciones variables del sistema;
La Figura 29 es un diagrama de entidad de datos que ilustra conexiones entre componentes de datos en el sistema;
La Figura 30 es un diagrama de entidad de datos que ilustra conexiones entre componentes de datos de almacenamiento de objetos en el sistema;
La Figura 31 es un diagrama de estructura de datos que ilustra los componentes de índice en el sistema;
50 La Figura 32 es un diagrama de entidad de datos que ilustra conexiones entre componentes de datos de almacenamiento de objetos en el sistema;
La Figura 33 es un diagrama de estructura de datos que ilustra los componentes del almacén de objetos en el sistema;
La Figura 34 es un diagrama de estructura de datos que ilustra los componentes del almacén de objetos en el sistema;
55 La Figura 35 es un diagrama esquemático de un asignador de vectores en el sistema;
La Figura 36 es un diagrama esquemático de un asignador de índices en el sistema;
La Figura 37 es un diagrama de relación de entidades de una red de objetos de ejemplo;
La Figura 38 es un diagrama de bloques de una arquitectura física de una unidad de procesamiento de gráficos (GPU);
60 La Figura 39 es un diagrama de flujo que ilustra un proceso de codificación de vectores;
La Figura 40 es un diagrama de flujo que ilustra un proceso de generación de huellas digitales;
La Figura 41 es un diagrama de flujo que ilustra un proceso de recuperación de similitudes; y
La Figura 42 es un diagrama de flujo de un proceso para unir un objeto a un centroide.

65

Descripción detallada

Algunas realizaciones implementan una red de inteligencia vectorial Simarray que incluye una red de agrupamiento de objetos asociativos de datos a datos con capacidades de coincidencia aproximada. En estas realizaciones, la inserción y eliminación de objetos Simarray (denominados en la presente descripción como "objetos"), así como la formación y destrucción de enlaces de objetos, se producen en tiempo real sin necesidad de indexación fuera de línea. Algunas realizaciones se basan en memoria para un funcionamiento de alto rendimiento. La durabilidad de los objetos se facilita a través de la escritura en disco de todas las operaciones de modificación de datos. La comparación de similitudes se logra a través de una combinación de funciones de filtración de huellas digitales vectoriales, indexación de dimensiones y modelo de espacio vectorial. Las redes de objetos se forman explícitamente mediante la aplicación de información de conectividad externa preexistente, e implícitamente mediante el análisis de similitud de objetos o grupos de objetos.

Los ejemplos de los métodos y sistemas discutidos en la presente descripción no se limitan en su aplicación a los detalles de construcción y la disposición de los componentes expuestos en la siguiente descripción o ilustrados en los dibujos adjuntos. Los métodos y sistemas pueden implementarse en otras realizaciones y practicarse o llevarse a cabo de diversas formas. En la presente descripción se proporcionan ejemplos de implementaciones específicas únicamente con fines ilustrativos y no pretenden ser limitativos. En particular, los actos, componentes, elementos y características descritas en relación con uno o más ejemplos no pretenden ser excluidos de un rol similar en cualquier otro ejemplo.

Además, la fraseología y la terminología que se usa en la presente descripción tienen fines descriptivos y no deben considerarse como limitantes. Cualquier referencia a ejemplos, realizaciones, componentes, elementos o actos de los sistemas y métodos a los que en la presente descripción se hace referencia en singular también puede abarcar realizaciones que incluyen una pluralidad, y cualquier referencia en plural a cualquier realización, componente, elemento o acto en este documento también puede abarcar realizaciones que incluyen sólo una singularidad. Las referencias en forma singular o plural no pretenden limitar los sistemas o métodos descritos actualmente, sus componentes, actos o elementos. El uso en la presente descripción de "incluye", "comprende", "tiene", "contiene", "involucra" y variaciones de los mismos pretende abarcar los elementos enumerados a continuación y sus equivalentes, así como elementos adicionales. Las referencias a "o" pueden interpretarse como inclusivas, de modo que cualquier término descrito usando "o" puede indicar uno solo, más de uno o todos los términos descritos. Además, en caso de usos inconsistentes de los términos entre este documento y los documentos incorporados en la presente descripción por referencia, el uso del término en las referencias incorporadas es complementario al de este documento; para inconsistencias irreconciliables, prevalece el uso del término en este documento.

Sistema informático

Como se analiza a continuación con respecto a las Figuras 1 y 29, varios aspectos y funciones descritos en la presente descripción pueden implementarse como componentes de hardware o software especializados que se ejecutan en uno o más sistemas informáticos. Hay muchos ejemplos de sistemas informáticos que están actualmente en uso. Estos ejemplos incluyen, entre otros, dispositivos de red, ordenadores personales, estaciones de trabajo, mainframes, clientes en red, servidores, servidores de medios, servidores de aplicaciones, servidores de bases de datos y servidores web. Otros ejemplos de sistemas informáticos pueden incluir dispositivos informáticos móviles, como teléfonos móviles y asistentes digitales personales, y equipos de red, como equilibradores de carga, enrutadores y conmutadores. Además, los aspectos pueden ubicarse en un solo sistema informático o pueden distribuirse entre una pluralidad de sistemas informáticos que se conectan a una o más redes de comunicaciones.

Por ejemplo, varios aspectos, funciones y procesos pueden distribuirse entre uno o más sistemas informáticos que se configuran para brindar un servicio a uno o más ordenadores clientes, o para realizar una tarea general como parte de un sistema distribuido. Además, los aspectos pueden realizarse en un sistema cliente-servidor o de varios niveles que incluye componentes distribuidos entre uno o más sistemas de servidor que realizan varias funciones. En consecuencia, las realizaciones no se limitan a la ejecución en ningún sistema o grupo de sistemas en particular. Además, los aspectos, funciones y procesos pueden implementarse en software, hardware o firmware, o cualquier combinación de los mismos. Por lo tanto, los aspectos, funciones y procesos pueden implementarse dentro de métodos, actos, sistemas, elementos del sistema y componentes usando una variedad de configuraciones de hardware y software, y los ejemplos no se limitan a ninguna arquitectura distribuida, red o protocolo de comunicación en particular.

Con referencia ahora a la Figura 1, se ilustra un diagrama de bloques de un sistema informático distribuido, en el que se practican varios aspectos y funciones. Como se muestra, el sistema informático distribuido incluye uno o más sistemas informáticos que intercambian información. Más específicamente, el sistema informático distribuido incluye los sistemas informáticos 102, 104 y 106. Como se muestra, los sistemas informáticos 102, 104 y 106 están interconectados y pueden intercambiar datos a través de una red de comunicación 108. La red 108 puede incluir cualquier red de comunicación a través de la cual los sistemas informáticos puedan intercambiar datos. Para intercambiar datos usando la red 108, los sistemas informáticos 102, 104 y 106 y la red 108 pueden usar varios métodos, protocolos y estándares, incluidos, entre otros, Canal de Fibra, Anillo Testigo, Ethernet, Ethernet Inalámbrico, Bluetooth, IP, IPV6, TCP/IP, UDP, DTN, HTTP, FTP, SNMP, SMS, MMS, SS7, JSON, SOAP, CORBA, REST y

servicios web. Para garantizar que la transferencia de datos sea segura, los sistemas informáticos 102, 104 y 106 pueden transmitir datos a través de la red 108 usando una variedad de medidas de seguridad que incluyen, por ejemplo, tecnologías SSL o VPN. Mientras que el sistema informático distribuido ilustra tres sistemas informáticos en red, el sistema informático distribuido no está tan limitado y puede incluir cualquier número de sistemas informáticos y dispositivos informáticos, que se conectan en red usando cualquier medio y protocolo de comunicación.

Como se ilustra en la Figura 1, el sistema informático 102 incluye un procesador 110, una memoria 112, un elemento de interconexión 114, una interfaz 116 y un elemento de almacenamiento de datos 118. Para implementar al menos algunos de los aspectos, funciones y procesos descritos en la presente descripción, el procesador 110 ejecuta una serie de instrucciones que dan como resultado datos manipulados. El procesador 110 puede ser cualquier tipo de procesador, multiprocesador o controlador. Además, aunque solo se muestra un único procesador, los expertos en la técnica apreciarán que en algunas realizaciones, las técnicas y los sistemas descritos en la presente descripción pueden implicar el uso de múltiples procesadores, en donde cada procesador puede tener uno o más núcleos. Los procesadores de ejemplo pueden incluir un procesador disponible comercialmente como un procesador Intel Xeon, Itanium, Core, Celeron o Pentium; un procesador AMD Opteron; un procesador Apple A4 o A5; un procesador Sun UltraSPARC; un procesador IBM Power5+; un chip de ordenador central de IBM; o un ordenador cuántico. Además, en algunas realizaciones, el procesador puede ser GPGPU. En aún otra realización, el procesador puede ser un procesador que se configura para ejecutar las extensiones AVX-512 discutidas anteriormente. El procesador 110 se conecta a otros componentes del sistema, incluidos uno o más dispositivos de memoria 112, mediante el elemento de interconexión 114.

La memoria 112 almacena programas (por ejemplo, secuencias de instrucciones codificadas para ser ejecutadas por el procesador 110) y datos durante el funcionamiento del sistema informático 102. Por lo tanto, la memoria 112 puede ser una memoria de acceso aleatorio, volátil, de rendimiento relativamente alto, tal como una memoria dinámica de acceso aleatorio ("DRAM") o una memoria estática ("SRAM"). Sin embargo, la memoria 112 puede incluir cualquier dispositivo para almacenar datos, como una unidad de disco u otro dispositivo de almacenamiento no volátil. Varios ejemplos pueden organizar la memoria 112 en estructuras particularizadas y, en algunos casos, únicas para realizar las funciones descritas en la presente descripción. Estas estructuras de datos pueden dimensionarse y organizarse para almacenar valores para datos y tipos de datos particulares.

Los componentes del sistema informático 102 se acoplan por un elemento de interconexión como el elemento de interconexión 114. El elemento de interconexión 114 puede incluir cualquier acoplamiento de comunicación entre los componentes del sistema, como uno o más buses físicos de acuerdo con tecnologías de bus de computación especializadas o estándar, como IDE, SCSI, PCI e InfiniBand. El elemento de interconexión 114 permite que las comunicaciones, incluidas las instrucciones y los datos, se intercambien entre los componentes del sistema del sistema informático 102.

El sistema informático 102 también incluye uno o más dispositivos de interfaz 116 tales como dispositivos de entrada, dispositivos de salida y dispositivos combinados de entrada/salida. Los dispositivos de interfaz pueden recibir entradas o proporcionar salidas. Más particularmente, los dispositivos de salida pueden generar información para presentación externa. Los dispositivos de entrada pueden aceptar información de fuentes externas. Los ejemplos de dispositivos de interfaz incluyen teclados, dispositivos de ratón, bolas de seguimiento, micrófonos, pantallas táctiles, dispositivos de impresión, pantallas de visualización, altavoces, tarjetas de interfaz de red, etc. Los dispositivos de interfaz permiten que el sistema informático 102 intercambie información y se comuniquen con entidades externas, como usuarios y otros sistemas.

El elemento de almacenamiento de datos 118 incluye un medio de almacenamiento de datos no volátil o no transitorio legible y grabable por ordenador en el que se almacenan instrucciones que definen un programa u otro objeto que se ejecuta por el procesador 110. El elemento de almacenamiento de datos 118 también puede incluir información que se registra, sobre o en el medio, y que se procesa por el procesador 110 durante la ejecución del programa. Más específicamente, la información puede almacenarse en una o más estructuras de datos que se configuran específicamente para conservar espacio de almacenamiento o aumentar el rendimiento del intercambio de datos. Las instrucciones pueden almacenarse de manera persistente como señales codificadas, y las instrucciones pueden hacer que el procesador 110 realice cualquiera de las funciones descritas en la presente descripción. El medio puede ser, por ejemplo, disco óptico, disco magnético o memoria flash, entre otros. En funcionamiento, el procesador 110 o algún otro controlador hace que los datos se lean desde el medio de grabación no volátil hacia otra memoria, como la memoria 112, que permite un acceso más rápido a la información por parte del procesador 110 que el medio de almacenamiento incluido en el elemento de almacenamiento de datos 118. La memoria puede ubicarse en el elemento de almacenamiento de datos 118 o en la memoria 112, sin embargo, el procesador 110 manipula los datos dentro de la memoria y luego copia los datos en el medio de almacenamiento asociado con el elemento de almacenamiento de datos 118 después de que se completa el procesamiento. Una variedad de componentes pueden administrar el movimiento de datos entre el medio de almacenamiento y otros elementos de memoria y los ejemplos no se limitan a componentes de administración de datos particulares. Además, los ejemplos no se limitan a un sistema de memoria o sistema de almacenamiento de datos en particular.

Aunque el sistema informático 102 se muestra a modo de ejemplo como un tipo de sistema informático en el que pueden practicarse varios aspectos y funciones, los aspectos y funciones no se limitan a implementarse en el sistema informático 102 como se muestra en la Figura 1. Pueden practicarse varios aspectos y funciones en uno o más ordenadores que tengan arquitecturas o componentes diferentes a los que se muestran en la Figura 1. Por ejemplo, el sistema informático 102 puede incluir hardware de propósito especial especialmente programado, tal como un circuito integrado de aplicación específica ("ASIC") adaptado para realizar una operación particular descrita en la presente descripción. Mientras que otro ejemplo puede realizar la misma función usando una cuadrícula de varios dispositivos informáticos de uso general que ejecutan MAC OS System X con procesadores Motorola PowerPC y varios dispositivos informáticos especializados que ejecutan hardware y sistemas operativos propietarios.

El sistema informático 102 puede ser un sistema informático que incluya un sistema operativo que gestione al menos una parte de los elementos de hardware incluidos en el sistema informático 102. En algunos ejemplos, un procesador o controlador, como el procesador 110, ejecuta un sistema operativo. Los ejemplos de un sistema operativo en particular que puede ejecutarse incluyen un sistema operativo basado en Windows, como Windows NT, Windows 2000 (Windows ME), Windows XP, Windows Vista o Windows 7, disponible en Microsoft Corporation, un MAC Sistema operativo OS System X o un sistema operativo iOS disponible de Apple Computer, una de las muchas distribuciones de sistemas operativos basados en Linux, por ejemplo, el sistema operativo Enterprise Linux disponible de Red Hat Inc., un sistema operativo Solaris disponible de Oracle Corporation, o a sistemas operativos UNIX disponibles de varias fuentes. Pueden usarse muchos otros sistemas operativos, y los ejemplos no se limitan a ningún sistema operativo en particular.

El procesador 110 y el sistema operativo juntos definen una plataforma informática para la cual se escriben programas de aplicación en lenguajes de programación de alto nivel. Estas aplicaciones componentes pueden ser ejecutables, intermedias, bytecode o código interpretado que se comunica a través de una red de comunicación, por ejemplo, Internet, usando un protocolo de comunicación, por ejemplo, TCP/IP. De manera similar, los aspectos pueden implementarse mediante un lenguaje de programación orientado a objetos, como .Net, SmallTalk, Java, C++, Ada, C# (C-Sharp), Python o JavaScript. También pueden usarse otros lenguajes de programación orientados a objetos. Alternativamente, pueden usarse lenguajes de programación funcionales, de secuencias de comandos o lógicos.

Además, varios aspectos y funciones pueden implementarse en un entorno no programado. Por ejemplo, los documentos creados en HTML, XML u otros formatos, cuando se ven en una ventana de un programa de navegador, pueden representar aspectos de una interfaz gráfica de usuario o realizar otras funciones. Además, varios ejemplos pueden implementarse como elementos programados o no programados, o cualquier combinación de los mismos. Por ejemplo, una página web puede implementarse usando HTML, mientras que un objeto de datos llamado desde la página web puede estar escrito en C++. Por lo tanto, los ejemplos no se limitan a un lenguaje de programación específico y podría usarse cualquier lenguaje de programación adecuado. En consecuencia, los componentes funcionales descritos en la presente descripción pueden incluir una amplia variedad de elementos (por ejemplo, hardware especializado, código ejecutable, estructuras de datos u objetos) que se configuran para realizar las funciones descritas en la presente descripción.

En algunos ejemplos, los componentes descritos en la presente descripción pueden leer parámetros que afectan las funciones realizadas por los componentes. Estos parámetros pueden almacenarse físicamente en cualquier forma de memoria adecuada, incluida la memoria volátil (como RAM) o la memoria no volátil (como un disco duro magnético). Además, los parámetros pueden almacenarse lógicamente en una estructura de datos adecuada (como una base de datos o un archivo definido por una aplicación de modo de usuario) o en una estructura de datos compartida comúnmente (como un registro de aplicación definido por un sistema operativo). Además, algunos ejemplos proporcionan interfaces de sistema y de usuario que permiten que entidades externas modifiquen los parámetros y, por lo tanto, configuren el comportamiento de los componentes.

Sistema de modelado de objetos

Diversas realizaciones utilizan uno o más sistemas informáticos que se configuran especialmente para modelar objetos y asociaciones entre objetos. La Figura 2 ilustra una de estas realizaciones, un sistema de modelado de objetos. Como se muestra, la Figura 2 incluye un modelador de red de objetos 202, un vectorizador 204, un almacén de datos de objetos externos 206 y una aplicación 208. El modelador de redes de objetos 202 incluye un gestor de objetos 210, un gestor de índices 212, un almacén de objetos 214, un almacén de índices 216 y un motor de consultas 218. El modelador de redes de objetos 202, el vectorizador 204, el almacén de datos de objetos externos 206 y la aplicación 208 pueden implementarse usando uno o más sistemas informáticos, como el sistema informático descrito anteriormente con referencia a la Figura 1. Como se ilustra en la Figura 2, el modelador de redes de objetos 202, el vectorizador 204, el almacén de datos de objetos externos 206 y la aplicación 208 intercambian (por ejemplo, envían o reciben) información a través de una o más conexiones 220. Las conexiones 220 pueden incluir cualquier dispositivo de comunicación a través del cual los componentes del ordenador puedan intercambiar información. Por ejemplo, las conexiones 220 pueden incluir conexiones de bus local, conexiones LAN, interfaces de programas de aplicación, enchufes y otros dispositivos de comunicación de datos que facilitan el flujo de información entre componentes informáticos.

En algunas realizaciones, la aplicación 208 se configura para interactuar con el modelador de redes de objetos 202 para explorar las relaciones entre los objetos descritos por el almacén de datos de objetos externos 206. Los ejemplos de la aplicación 208 incluyen aplicaciones que obtienen valor de la identificación y exploración de asociaciones entre objetos, tales como tesauros, diccionarios, enciclopedias, sistemas de extracción de datos, sistemas de inteligencia empresarial y motores de búsqueda de Internet, entre otros sistemas. En estas realizaciones, la aplicación 208 interactúa con el modelador de redes de objetos 202 a través de una interfaz expuesta por el modelador de redes de objetos 202, como una interfaz de programa de aplicación (API). A través de la API, la aplicación 208 transmite varios mensajes al modelador de redes de objetos 202. Estos mensajes pueden incluir solicitudes para que el modelador de redes de objetos 202 complete un conjunto de estructuras de datos, como las estructuras de datos descritas más adelante con referencia a la Figura 29. Los mensajes también pueden incluir solicitudes para que el modelador de redes de objetos 202 ejecute consultas contra estas estructuras de datos. En algunas realizaciones, el modelador de redes de objetos 202 se configura para procesar estos mensajes API en respuesta a su recepción. De esta forma, el modelador de redes de objetos 202 admite el funcionamiento de la aplicación 208 al crear, mantener y consultar redes de objetos como se describe más adelante.

En una realización que se ilustra por la Figura 2, el almacén de datos de objetos externos 206 incluye información descriptiva de los objetos externos seleccionados para el análisis por la aplicación 208. Ejemplos de estos objetos externos incluyen palabras, conceptos, personas, productos y documentos, entre otros objetos. El almacenamiento de datos de objetos externos 206 puede implementarse en forma de cualquier construcción lógica capaz de almacenar información en un medio legible por ordenador que incluye archivos planos, archivos indexados, bases de datos jerárquicas, bases de datos relacionales o bases de datos orientadas a objetos.

En otras realizaciones, el vectorizador 204 genera vectores a partir de los datos sin procesar almacenados en el almacén de datos de objetos externos, que es un proceso conocido en la técnica como vectorización. Los vectores que se generan por el vectorizador 204 incluyen información descriptiva de los objetos externos. Un buen vectorizador puede extraer características clave de los datos sin procesar, creando así un vector disperso en un espacio dimensional alto. Un tipo común de vectorizador es el que extrae palabras clave y frases importantes (ngramas) de un texto libre. Otros vectorizadores pueden generar vectores que representan imágenes, medios o cualquier otra forma de información como un conjunto de características.

De acuerdo con algunas realizaciones, el gestor de objetos 210 se configura para crear, actualizar y eliminar objetos que se procesan por el modelador de redes de objetos 202. En estas realizaciones, el gestor de objetos 210 recibe vectores externos del vectorizador 204, codifica los vectores externos como vectores internos, resume los vectores internos en huellas digitales y usa esta información para mantener el almacenamiento de objetos 214. En algunas realizaciones, el gestor de objetos 210 ejecuta estas acciones en respuesta a la recepción de un mensaje que solicita las acciones. El gestor de objetos puede recibir dicho mensaje de la aplicación 208, el vectorizador 204, un archivo de configuración o de alguna otra fuente. Más adelante se describen ejemplos de los procesos que el gestor de objetos 210 se configura para realizar.

En algunas realizaciones, el gestor de índices 212 se configura para crear, actualizar y eliminar índices almacenados en el almacén de índices 216. En estas realizaciones, el gestor de índices 212 crea y actualiza índices en tiempo real a medida que los objetos se almacenan o modifican en el almacén de objetos 214. Más adelante se describen ejemplos de los procesos realizados y los datos manipulados por el gestor de índices 212.

En algunas realizaciones, el almacén de objetos 214 se configura para almacenar objetos creados y manipulados por el gestor de objetos 210. En estas realizaciones, el almacén de objetos 214 incluye estructuras de datos especialmente adaptadas para minimizar la fragmentación de la memoria y maximizar el rendimiento. Más adelante se describen ejemplos de las estructuras y datos incluidos en el almacén de objetos 214.

En algunas realizaciones, el almacén de índices 216 se configura para almacenar índices creados y manipulados por el gestor de índices 212. En estas realizaciones, el almacén de índices 216 incluye estructuras de datos especialmente adaptadas para minimizar la fragmentación de la memoria y maximizar el rendimiento. Más adelante se describen ejemplos de las estructuras y datos incluidos en el almacén de índices 216.

En algunas realizaciones, el motor de consulta 218 se configura para procesar mensajes de proceso de consulta que incluyen solicitudes de consulta. Estos mensajes de consulta pueden incluir información descriptiva de uno o más objetos de sondeo para los que se buscan objetos asociados. En respuesta a la recepción de un mensaje de consulta, el motor de consulta 218 procesa el mensaje de consulta. Los ejemplos de los procesos y datos manipulados por el motor de consulta 218 se describen más adelante.

Objetos Simarray y Redes de Objetos

En varias realizaciones, el gestor de objetos se configura para implementar una red de objetos. Esta red de objetos puede incluir uno o más objetos que representan entidades del mundo real, como documentos, personas, productos, páginas web, etc. Cada uno de los objetos dentro de la red de objetos puede incluir un conjunto de atributos que

describen una entidad del mundo real representada por el objeto. La Tabla 1 enumera varios ejemplos de estos atributos:

TABLA 1

Atributo	Descripción	Comentario
OBID	Identificador único que permite la recuperación directa del objeto a través del <i>índice de objetos</i> .	Múltiples manifestaciones del mismo objeto comparten el mismo OBID pero tienen diferentes SUBID.
SUBID	Referencia de salida a otro objeto que permite la recuperación del objeto referenciado a través del <i>índice del objeto</i> y la recuperación de referentes entrantes a través del <i>índice inverso</i> .	Puede ser NULO. Un objeto sin referencia de salida (SUBID=NULO) puede denominarse en la presente descripción como TERMINAL. Un objeto con una o más referencias de salida es un objeto DIRIGIDO.
VECTOR	Conjunto de características descriptivas y ponderadas de un objeto que ubican el objeto en un espacio de alta dimensión. Forma la base para la coincidencia aproximada entre objetos en la red u objetos de sondeo. Permite la búsqueda de similitud de recuperación máxima a través de los <i>índices de dimensión directa</i> .	Vector disperso en espacio de alta dimensión.
HUELLA DIGITAL (FP)	Un hash sensible a la localidad de VECTOR. Este es un vector de bits de dimensionalidad reducida derivado del conjunto de características del objeto (VECTOR). Permite una búsqueda de similitud de alto rendimiento a través de los <i>índices de permutación</i> .	Vector denso en espacio binario de 64 dimensiones.
CARDINALIDAD	Un contador que refleja el número de referencias entrantes a un objeto desde otros objetos.	
BLOB	Cualquier metadato adicional para asociar con el objeto en forma de información binaria arbitraria de cualquier tamaño hasta 2 GB por objeto.	Puede ser NULO. La semántica BLOB se aplica solo a capas de aplicación de nivel superior.

En algunas realizaciones, el gestor de objetos asocia objetos entre sí dentro de una red de objetos mediante, por ejemplo, referencias SUBID. En estas realizaciones, el gestor de objetos atraviesa objetos dentro de la red de objetos mediante, por ejemplo, un *índice de objetos* y un *índice inverso*. El gestor de objetos puede asociar objetos en al menos dos formas distintas: (i) explícitamente a partir de enlaces derivados del mundo real (por ejemplo, aprovechando las relaciones previamente establecidas) e (ii) implícitamente desde la "naturaleza del objeto" a través del análisis de similitud.

Por ejemplo, el gestor de objetos puede crear una red de objetos que asocie el objeto A (**A**) con el objeto B (**B**) instanciando dos objetos con los siguientes atributos:

Objeto A		Objeto B	
OBID	A	OBID	B
SUBID	B	SUBID	NULO

El gestor de objetos no necesita registrar un motivo por el cual **A** está asociado con **B** para que exista la red de objetos, pero el gestor de objetos puede registrar el motivo internamente como un atributo LNK que tiene un valor de:

LNK = IMPLÍCITO **A** → **B** debido a la "naturaleza del objeto", es decir, **A** y **B** son similares
LNK = EXPLÍCITO **A** → **B** debido al conocimiento previo externo de las relaciones de objeto
LNK = DUAL **A** → **B** debido a la relación externa, así como a la similitud

Por lo tanto, en un ejemplo, el gestor de objetos crea una representación mínima para que **A** se asocie explícitamente con **B** debido al conocimiento previo externo como:

Objeto A		Objeto B	
OBID	A	OBID	B
SUBID	B	SUBID	NULO
LNK	EXPL	LNK	NULO

En algunas realizaciones, el gestor de objetos registra asociaciones implícitas o duales de manera similar (por ejemplo, usando un atributo LNK). Sin embargo, el descubrimiento de asociaciones implícitas o duales requiere un análisis de similitud usando vectores y huellas digitales, como se describe anteriormente. El gestor de objetos puede crear una

representación mínima de **A** asociada implícitamente con **B** debido a la coincidencia aproximada de las características del objeto como:

Objeto A		Objeto B	
OBID	A	OBID	B
SUBID	B	SUBID	NULO
LNK	EXPL	LNK	NULO
VECTOR	[x,y,z,....]	VECTOR	[x,y,.....]
FP	11001010...	FP	11011010.....

Alternativamente, el gestor de objetos puede crear una representación mínima de **A** asociada implícita y explícitamente con **B** como:

Objeto A		Objeto B	
OBID	A	OBID	B
SUBID	B	SUBID	NULO
LNK	AMBOS	LNK	NULO
VECTOR	[x,y,z,....]	VECTOR	[x,y,.....]
FP	11001010...	FP	11011010.....

La Figura 3 ilustra adicionalmente ejemplos de asociaciones de objetos que el gestor de objetos se configura para crear en función de las relaciones entre los objetos. Como se ilustra en la Figura 3, estas asociaciones incluyen una asociación implícita unidireccional, una asociación explícita unidireccional, una asociación explícita e implícita unidireccional, una asociación implícita bidireccional, una asociación explícita bidireccional, una asociación explícita e implícita bidireccional y una asociación NULA. La Figura 3 ilustra cada una de estas asociaciones usando tanto representaciones gráficas como tablas de objetos, como se muestra.

En otras realizaciones, el gestor de objetos se configura para instanciar objetos que tienen tipos de vectores (VEC). La Figura 4 ilustra ejemplos de estos tipos de vectores. Como se ilustra en la Figura 4, los tipos de vectores incluyen un tipo de vector estándar, un tipo de vector centroide y un tipo de vector NULO. La Figura 4 ilustra cada uno de estos tipos de vectores usando tanto representaciones gráficas como tablas de objetos, como se muestra.

Como se describió anteriormente, los componentes del modelador de redes de objetos utilizan tipos de vector y vectores para descubrir asociaciones implícitas de objetos. Se aprecia que la asociación explícita no requiere que un objeto tenga un valor de vector o un valor de tipo de vector y, por lo tanto, un objeto puede tener un vector sin elementos y un valor de tipo de vector NULO. La Figura 5 ilustra un ejemplo de una red de objetos que puede crearse y gestionarse por el gestor de objetos. Como se muestra en la Figura 5, la red de objetos incluye los objetos centroides **CA**, **CB** y **CD**. **CA** también es un objeto TERMINAL y no incluye asociaciones con otros objetos. **CB** incluye una asociación bidireccional explícita con **CD** y **CD** incluye una asociación bidireccional explícita con **CB**.

En el ejemplo de la Figura 5, cada uno de los objetos **A1**, **A2** y **A3** es un objeto de vector estándar que incluye una asociación unidireccional implícita con **CA**. **A3** incluye además una asociación unidireccional implícita con **B3**, una asociación bidireccional dual con **B2** y una asociación unidireccional explícita con **N1**. **N1** es un objeto vector NULO que incluye asociaciones unidireccionales explícitas con **N2** y **N3**. Tanto **N2** como **N3** son vectores NULOS, objetos TERMINALES. Cada uno de los objetos **B1**, **B2**, **B3** y **B4** es un objeto vectorial estándar que incluye una asociación unidireccional implícita con **CB**.

Como se muestra en la Figura 5, cada uno de los objetos **D1**, **D2**, **D3**, **D4**, **D5** y **D6** es un objeto vector estándar que incluye una asociación unidireccional implícita con **CB**. **D6** también incluye una asociación unidireccional explícita con **N4**, una asociación unidireccional dual con **N5** y una asociación unidireccional implícita con **N6**. **N4** es un vector NULO, objeto TERMINAL. Tanto **N5** como **N6** son objetos vectoriales estándar, TERMINAL.

En algunas realizaciones, el gestor de objetos se configura para mantener objetos centroides que representan un grupo o grupo de otros objetos. En estas realizaciones, un vector de un objeto centroide es una función de los vectores de los objetos que se unen al objeto centroide a través de asociaciones unidireccionales o bidireccionales. Por ejemplo, el vector de un centroide puede incluir las dimensiones de los objetos que se unen al centroide (hasta el límite de truncamiento) ordenados por magnitud promedio). Para facilitar el mantenimiento de objetos centroides por parte de otros componentes (por ejemplo, componentes de la capa de aplicación), el gestor de objetos puede exponer una interfaz de clúster a través de la cual el gestor de objetos recibe mensajes de agrupación de los otros componentes. Estos mensajes de agrupación pueden incluir instrucciones para crear un centroide, unir un objeto a un centroide o separar un objeto de un centroide, entre otras instrucciones de agrupación.

En algunas realizaciones, el gestor de objetos se configura para ejecutar instrucciones de agrupación incluidas dentro de los mensajes de agrupación que se reciben a través de la interfaz de agrupación. Por ejemplo, en al menos una realización, el gestor de objetos se configura para responder a un mensaje de agrupación que solicita la creación de un centroide. Este tipo de mensaje de agrupamiento puede incluir metadatos descriptivos del centroide (por ejemplo,

ID de grupo, nombre, etc.). En respuesta a la recepción de este tipo de mensaje de agrupamiento, el gestor de objetos instancia un objeto de centroide de vector NULO al que pueden unirse objetos. Además, en estas realizaciones, el gestor de objetos se configura para mantener automáticamente los recuentos de vector y cardinalidad del centroide (referencias entrantes al centroide) a medida que los objetos se adjuntan o desconectan en tiempo real en respuesta a mensajes de agrupación.

Las Figuras 6 a la 8 ilustran un proceso de agrupamiento que el gestor de objetos se configura para ejecutar en algunas realizaciones. El proceso de agrupamiento adjunta un nuevo objeto sin enlaces a un centroide dentro de una red de objetos existente, como la red de objetos que se ilustra en la Figura 6, donde las asociaciones se definen implícitamente en función del contenido del objeto (por ejemplo, el texto del documento). Como se muestra en la Figura 6, el proceso de agrupación incluye los actos de enviar una consulta basada en el nuevo objeto, recibir los resultados de la consulta, identificar el centroide y unir el nuevo objeto al centroide. Como se muestra en la Figura 6, la red de objetos existente incluye el centroide C y los objetos **A1**, **A2** y **A3**.

Refiriéndose brevemente a la Figura 42, la Figura 42 ilustra un diagrama de flujo de un proceso para unir un objeto a un centroide en línea con el proceso que se muestra en las Figuras 6 a la 8. El gestor de objetos comienza el proceso de agrupamiento 4200 en 4202. En el acto 4204, el gestor de objetos envía un mensaje de consulta al motor de consultas. El mensaje de consulta puede incluir una métrica de similitud de umbral y el vector del nuevo objeto (por ejemplo, el objeto **A4** ilustrado en la Figura 7), o una parte del mismo. En respuesta a la recepción del mensaje de consulta, el motor de consulta ejecuta una consulta de vector basada en el mensaje de consulta y devuelve un mensaje de resultados que incluye resultados de consulta que identifican uno o más objetos (por ejemplo, el objeto **A2** ilustrado en la Figura 8) que cumplen con la métrica de similitud de umbral. En el acto 4206, el gestor de objetos recibe el mensaje de resultados. En el acto 4208, el gestor de objetos identifica un centroide (por ejemplo, el objeto **C** ilustrado en la Figura 8) vinculado directa o indirectamente a un objeto identificado en los resultados de la consulta como similar (por ejemplo, el más similar) al nuevo objeto. En algunas realizaciones y como se ilustra en la Figura 8, el gestor de objetos identifica el centroide atravesando la red de objetos desde el objeto identificado como similar (por ejemplo, el objeto **A2**) hasta que se alcanza el centroide. En el acto 4210, el gestor de objetos adjunta un nuevo objeto al centroide identificado actualizando los atributos del nuevo objeto y la información incluida en los diversos índices. Dentro del acto 4210, el gestor de objetos también actualiza la cardinalidad y el vector del centroide en función de la existencia y el vector del nuevo objeto. El gestor de objetos finaliza el proceso de agrupamiento 4200 en 4212.

Se aprecia que la ejecución de un proceso de agrupamiento de acuerdo con el proceso de agrupamiento 4200 puede crear grandes agrupamientos con elementos que tienen poco contenido en común y donde, en consecuencia, el centroide está relativamente distante de los objetos vinculados. Para abordar este problema, los componentes que interactúan con el gestor de objetos pueden configurarse para monitorear la dispersión permitida dentro de un clúster (por ejemplo, mediante el seguimiento de una o más métricas de dispersión que indican la cantidad de dispersión) y volver a agrupar un clúster en clústeres más pequeños para limitar la cantidad de dispersión donde las métricas de dispersión transgreden (por ejemplo, superan) un umbral de dispersión.

En algunas realizaciones, el gestor de objetos se configura para procesar mensajes de agrupación que solicitan asociaciones explícitas entre objetos. Este tipo de mensaje de agrupamiento puede incluir información que identifique los objetos a unir explícitamente y el tipo de asociación a establecer (por ejemplo, una asociación unidireccional o una asociación bidireccional). En respuesta a la recepción de un mensaje de agrupación que solicita una asociación explícita, el gestor de objetos ajusta los atributos de los objetos identificados y los índices para implementar la asociación solicitada.

En algunas realizaciones, los conjuntos de grupos se crean asociando centroides con otros centroides. Por ejemplo, un grupo de páginas web puede asociarse con otro grupo de páginas web, creando así un supergrupo en el que los subgrupos están implícitamente vinculados internamente y explícitamente vinculados a otros subgrupos. De acuerdo con diversas realizaciones, el gestor de objetos se configura para asociar explícita o implícitamente un centroide en respuesta a la detección de una o más condiciones. Por ejemplo, en una realización, el gestor de objetos se configura para asociar un primer centroide con un segundo centroide en respuesta a la detección de que más de un número umbral de objetos vinculados al primer centroide tienen asociaciones explícitas con objetos vinculados al segundo centroide. Se aprecia que, en algunas realizaciones, el gestor de objetos se configura para representar relaciones más débiles simplemente asociando objetos individuales vinculados a centroides, en lugar de asociar los propios centroides.

La Figura 9 ilustra un ejemplo de un súper clúster en el que el centroide **C_A** incluye una asociación bidireccional explícita con el centroide **C_B** y el centroide **C_B** incluye una asociación bidireccional explícita con **C_A**. Al ejecutar consultas de vectores para un objeto de sondeo contra la red de objetos que se ilustra en la Figura 9, si el motor de consulta determina que el objeto de sondeo es similar a cualquiera de los objetos **A1**, **A2**, **A3** o **A4**, el motor de consulta también probará la similitud entre el objeto de sondeo y cada uno de los objetos **B1**, **B2** y **B3** debido al enlace bidireccional explícito entre **C_A** y **C_B**.

La Figura 10 ilustra otro ejemplo de un súper clúster en el que el subcluster con el centroide **C_A** está débilmente vinculado con el subcluster con el centroide **C_D** a través de una asociación unidireccional explícita con **D2** incluida en

A3. Al ejecutar consultas de vectores para un objeto de sondeo contra la red de objetos que se ilustra en la Figura 10, si el motor de consulta determina que el objeto de sondeo es similar a cualquiera de los objetos **A1**, **A2**, **A3** o **A4**, el motor de consulta también probará la similitud entre el objeto de sondeo y cada uno de los objetos **B1**, **B2** y **B3** debido del enlace bidireccional explícito entre **C_A** y **C_B**. Además, si el motor de consultas determina que el objeto de sondeo es similar a **A3**, el motor de consultas también probará la similitud entre el objeto sondeo y **D1**, **D2** y **D3**.

La Figura 11 ilustra otro ejemplo de una red de objetos producida por el gestor de objetos. Como se muestra en la Figura 11, cada objeto dentro de la red de objetos es un objeto vector NULO que enumera su cardinalidad entre paréntesis. Por ejemplo, **P1** tiene una cardinalidad de 0, mientras que **P12** tiene una cardinalidad de 1. En algunas realizaciones, el gestor de objetos mantiene internamente una manifestación para cada asociación de salida de un objeto. Por ejemplo, el objeto P3 tiene cuatro manifestaciones internas (P31, P32, P33 y P34), cada una con metadatos opcionales que sirven como "texto de anclaje" para la asociación de salida, fuerza de asociación opcional basada en una variedad de factores y rango general opcional en relación con otros objeto. En algunas realizaciones, los componentes que interactúan con el gestor de objetos ejecutan un análisis de asociación de saltos múltiples analizando las asociaciones de entrada de un objeto de destino, los objetos vinculados al objeto de destino a través de las asociaciones de entrada, las asociaciones de entrada de esos objetos, etc., con una profundidad arbitraria determinada por el componente que interactúa con el gestor de objetos.

En algunas realizaciones, el gestor de objetos se configura para atravesar la red de objetos tanto hacia adelante como hacia atrás usando los índices de objeto e inverso. Por ejemplo, un objetivo de una aplicación puede ser mejorar la clasificación de relevancia en un motor de búsqueda. Modelar el gráfico de enlaces de páginas indexadas por adelantado en el modelador de redes de objetos permite que la aplicación mejore el motor de búsqueda con información de enlaces. Antes de inyectar una página en el motor de búsqueda, la aplicación puede consultar el modelador de red de objetos precargado en tiempo real para extraer información sobre la posición de la página y su importancia en la red usando un análisis hacia delante/hacia atrás arbitrariamente complejo, así como cualquier rango de página estática o texto de anclaje de los referentes entrantes para indexar junto con el texto original de la página.

Ir más allá del gráfico de enlace estándar al incluir asociaciones implícitas entre páginas amplía enormemente las capacidades de la aplicación. Surgirán relaciones difusas que permitirán la vinculación de conceptos, la eliminación de duplicados, el contenido relacionado, etc.

Codificación de vectores

En algunas realizaciones, el motor de consulta se configura para ejecutar coincidencias parciales basadas en un *modelo de espacio vectorial*. En estas realizaciones, el gestor de objetos se configura para recibir vectores externos del vectorizador y generar vectores internos basados en estos vectores externos recibidos. Al generar vectores internos, el gestor de objetos se da cuenta de los diversos beneficios que incluyen la eficiencia del procesamiento de datos, la eficiencia del almacenamiento y la uniformidad. Por ejemplo, cuando el vectorizador se aplica a texto libre, generará vectores externos con dimensiones (características) que son cadenas de longitud arbitraria. Para lograr uniformidad y eficiencia, el gestor de objetos codifica el vector como se describe más adelante.

En algunas realizaciones, un vector externo V_{ext} tiene una forma general que puede expresarse como la Ecuación 1:

$$V_{ext} = \{(característica_1, peso_1), (característica_2, peso_2), \dots\} \quad (1)$$

donde $(característica_n, peso_n)$, es el n -ésimo *elemento del vector*, ordenado por peso en orden descendente. Un vector externo puede contener cualquier número de elementos. Cada *elemento* puede incluir una $característica_n$, que son datos que describen el objeto del que se deriva y puede ser de cualquier tamaño, y $peso_n$, que es cualquier número positivo real que denota la importancia de la $característica_n$ en relación con otras características.

El gestor de objetos convierte V_{ext} a una forma interna V_{int} a través de hash de características, la normalización de peso y la cuantificación de longitud. En estas realizaciones, el gestor de objetos se configura para generar un vector interno, V_{int} con una forma general que puede expresarse como la Ecuación 2:

$$V_{int} = \{(dim_1, mag_1), (dim_2, mag_2), \dots\} \quad (2)$$

donde el elemento (dim_n, mag_n) se deriva de $(característica_n, peso_n)$. El gestor de objetos puede almacenar cualquier número de elementos en un vector interno, hasta un límite de truncamiento específico de la implementación m . Por lo tanto, en algunas realizaciones, el gestor de objetos se configura para trunca, descartar y no incluye dentro de V_{int} ninguna $(característica_n, peso_n)$ donde $n > m$. De acuerdo con varias realizaciones, el gestor de objetos almacena V_{int} internamente como una matriz de números enteros de 32 bits, cada uno de los cuales representa un elemento que usa 26 bits para dim_n y 6 bits para mag_n , como se ilustra en la Figura 12.

En algunas realizaciones, el gestor de objetos se configura para cuantificar el tamaño Q_k de la matriz interna en función del número de elementos (distintos de cero) en V_{int} . En estas realizaciones, el gestor de objetos se configura para asignar la matriz que contiene V_{int} del *asignador de vector* k -ésimo y establece los elementos finales (es decir,

elementos sin característica correspondiente en el vector externo) establecidos en cero. Usando $n(X)$ para indicar el número de elementos en X , la relación entre el límite de truncamiento, el tamaño de la matriz interna y V_{int} puede expresarse como Desigualdad 3:

$$m \geq Q_k \geq n(V_{int}) \quad (3)$$

Para fines de eficiencia de almacenamiento, el gestor de objetos se configura para establecer asignaciones de memoria personalizadas para las matrices vectoriales, cada una de las cuales produce una matriz int32 de tamaño Q_k que es lo suficientemente grande como para almacenar V_{int} . A menos que $n(V_{int}) = Q_k$, el gestor de objetos establece los elementos finales de la matriz de vectores en cero.

En algunas realizaciones, el conjunto de asignadores de vectores A se cuantifica para evitar la sobrecarga de tener un asignador separado para cada tamaño de vector posible $n(V_{int}) \leq m$, equilibrado con la necesidad de minimizar el número de elementos cero finales desperdiciados cuando $n(V_{int}) < Q_k$. En estas realizaciones, el gestor de objetos selecciona el asignador A_k más pequeño para generar una matriz con capacidad suficiente para almacenar un vector interno.

En algunas realizaciones, el conjunto de tamaños de asignador Q es un parámetro configurable que puede ajustarse en tiempo de compilación o tiempo de ejecución para adaptarse a una o más aplicaciones, como la aplicación 208 descrita anteriormente con referencia a la Figura 2. El conjunto de tamaños del asignador Q puede expresarse como la Ecuación 4:

$$Q = \{ Q_1, Q_2, \dots, Q_k, \dots, Q_{\max} \}, \quad Q_{\max} = m \quad (4)$$

Las Figuras 13 y 39 ilustran, en conjunto, un proceso de conversión para crear V_{int} a partir de V_{ext} , que se ejecuta por el gestor de objetos de acuerdo con algunas realizaciones. Con respecto a la Figura 39, la Figura 39 es un diagrama de flujo que ilustra un proceso de codificación de vectores 3900 que incluye actos de recibir un vector externo, seleccionar un asignador de vectores, generar representaciones internas, normalizar pesos y cuantificar longitudes. El proceso de conversión 3900 comienza en 3902.

En el acto 3904, el gestor de objetos recibe un vector externo para su procesamiento. El vector externo puede transmitirse desde el vectorizador. Como se muestra en la Figura 13, el vector externo puede incluir un conjunto de elementos $\{característica_1, \dots, característica_n\}$ donde cada elemento incluye una característica y un peso.

En el acto 3906, el gestor de objetos asigna una matriz de vectores internos para almacenar V_{int} . En una realización, el gestor de objetos analiza el vector externo para determinar su tamaño y selecciona un asignador de vectores (ilustrado como A_k en la Figura 13) tal que $Q_k \geq n(V_{ext})$, $Q_k \leq m$. A continuación, el gestor de objetos asigna una nueva matriz de vectores internos usando A_k (consulte "Vector interno V_{int} " en la Figura 13).

En el acto 3908, el gestor de objetos crea una representación interna de cada característica incluida en el vector externo. En una realización, el gestor de objetos crea representaciones internas codificando la $característica_n$ como un número entero dim_n usando una función hash: $dim_n = hash(característica_n)$ para todo $n > Q_k$. En algunas realizaciones, hash() es rápido y produce números enteros distribuidos uniformemente de un tamaño predeterminado (por ejemplo, 26 bits). En el acto 3910, el gestor de objetos normaliza los pesos de las representaciones internas creadas en el acto 3908. En una realización, el gestor de objetos codifica cada $peso_n$ como mag_n usando una función de escala y normalización tal que el peso máximo se asigna a un máximo predefinido (por ejemplo, $2^6 - 1 = 63$) y el peso mínimo se asigna a un mínimo predefinido (por ejemplo, 1): $mag_n = escala(peso_n)$ para todo $n > Q_k$. Este acto puede verse facilitado por la prenormalización de V_{ext} para representar el peso de característica máximo posible como una cantidad conocida, como 1,0.

En el acto 3912, el gestor de objetos cuantifica la longitud de V_{int} . En una realización, el gestor de objetos establece $dim_n = 0$, $mag_n = 0$ para todo $n > Q_k$, como se ilustra en el elemento final de la matriz de vectores internos en la Figura 13.

En algunas realizaciones, no se requiere que el vector de características V_{ext} se almacene dentro del modelador de red de objetos 202, pero puede incluirse como parte de los metadatos del objeto. En estas realizaciones, el vector interno V_{int} se almacena en el objeto que el vector describe como un puntero a la matriz int32 devuelta por el asignador A_k . Múltiples manifestaciones $O_1, O_2, O_3, \dots$ del mismo objeto O (mismo OBID, diferente SUBID) pueden compartir la misma instancia de vector al hacer referencia a la misma matriz en la memoria. En estas realizaciones, el gestor de objetos se configura para, tras la eliminación de una manifestación, determinar si las manifestaciones adicionales hacen referencia a la matriz y, si no, desasignar la matriz. Por ejemplo, en las realizaciones en las que se usa un recuento de referencia para rastrear el número de manifestaciones que hacen referencia a la matriz, el gestor de objetos puede disminuir el recuento de referencia durante la eliminación de una manifestación de objeto. Además, cuando el recuento de referencias es igual a cero, el gestor de objetos puede desasignar la matriz.

Si bien el espacio de características de V_{ext} es prácticamente infinito, el espacio de dimensiones de V_{ext} está aún más restringido como parte de la codificación. En algunas realizaciones, debido al uso de 26 bits para dimensiones vectoriales internas, el modelador de redes de objetos opera en un espacio vectorial finito, dimensional de $2^{26} - 1 = 67\,103\,863$. ($dim=0$ se reserva para matrices de vectores de relleno cero donde $n(V_{int,i}) < Q_k$.)

Se aprecia que, a pesar de que 67 millones de grados de libertad inicialmente pueden parecer abundantes, pueden ocurrir solapes (colisiones) incluso dentro de un vocabulario de características limitado que es de órdenes de magnitud menos diversa que el espacio vectorial del modelo de objeto, debido a un fenómeno comúnmente conocido como el problema del cumpleaños. En algunas realizaciones, el solape de dimensión puede dar lugar a falsos positivos ocasionales al comparar vectores, lo que afecta a la precisión general. Sin embargo, en estas realizaciones, las ventajas de rendimiento de los vectores internos compactos superan las desventajas de los falsos positivos esporádicos. Además, para una aplicación que requiere una precisión absoluta, algunas realizaciones incluyen una configuración de gestor de objetos para almacenar V_{ext} a lo largo del objeto como metadatos para el filtrado (antisolapamiento) como se describe más adelante.

Medida de similitud vectorial

En algunas realizaciones, el motor de consulta se configura para determinar asociaciones implícitas (coincidencia aproximada) usando el cálculo de similitud de vectores. Para ilustrar esta configuración dentro del contexto de dos dimensiones, el motor de consulta puede calcular la similitud de dos vectores **A** y **B** como se expresa en la Ecuación 5:

$$\text{Coseno}(\mathbf{A}, \mathbf{B}) = \theta = \frac{\mathbf{A} \cdot \mathbf{B}}{|\mathbf{A}| |\mathbf{B}|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}} \quad (5)$$

El ángulo θ representa la alineación espacial, como se ilustra en la Figura 14, y no tiene en cuenta las diferencias de longitud entre **A** y **B**. Este enfoque se denomina como similitud de coseno y puede aplicarse a cualquier número de dimensiones.

En otras realizaciones, el motor de consulta se configura para medir la similitud usando el cálculo expresado en la Ecuación 6:

$$\text{Jaccard}(\mathbf{A}, \mathbf{B}) = \frac{A \cap B}{A \cup B} = \frac{\sum_{s_j \in A \cap B} \min(\text{freq}_A(s_j), \text{freq}_B(s_j))}{\sum_{s_j \in A \cup B} \max(\text{freq}_A(s_j), \text{freq}_B(s_j))} \quad (6)$$

Este enfoque se conoce como el índice de Jaccard. El índice de Jaccard para dos conjuntos **A** y **B** se define como la relación intersección/unión y mide la superposición entre las dimensiones del vector; cuanto más se superponen, mayor es la similitud.

Algunas realizaciones descritas en la presente descripción manifiestan una apreciación de que ni la similitud del coseno ni el índice de Jaccard son ideales para cada circunstancia. En estas realizaciones, el motor de consulta se configura para combinar los dos en una medida compuesta que ayuda a contrarrestar el comportamiento idiosincrático no deseado en uno u otro. Este enfoque combinado puede expresarse como la Ecuación 7:

$$\text{Sim}(\mathbf{A}, \mathbf{B}) = \text{Coseno}(\mathbf{A}, \mathbf{B})^\alpha \cdot \text{Jaccard}(\mathbf{A}, \mathbf{B})^\beta \quad (7)$$

Los exponentes α y β son parámetros variables configurables de 0,0 a 1,0 (como $\alpha + \beta = 1,0$) para adaptarse a aplicaciones en las que una medida de similitud es más adecuada que la otra.

Se aprecia que la maldición de la dimensionalidad pone un límite a la eficiencia computacional, incluso cuando se usan vectores internos compactos. Por esta razón, las realizaciones del modelador de redes de objetos no se basan únicamente en la comparación de vectores de fuerza bruta para la coincidencia aproximada. Algunas realizaciones utilizan índices de dimensión directos en el vector interno para los primeros **M** elementos. En estas realizaciones, el motor de consultas garantiza la recuperación de vectores que comparten al menos una dimensión entre los **M** elementos superiores. Sin embargo, se aprecia que configurar la indexación de dimensión directa para todos los elementos tiene un impacto en el rendimiento debido a la naturaleza $O(n^2)$ de este enfoque. Las realizaciones abordan este problema usando la toma de huellas digitales, que se describe más adelante.

Huellas digitales

El modelador de redes de objetos se configura para abordar la maldición de la dimensionalidad para la comparación de vectores mediante una técnica de reducción de la dimensionalidad conocida como huellas digitales basadas en *hash sensible a la localidad* (LSH). En estas realizaciones, el modelador de redes de objetos usa un enfoque en el que varios fragmentos de LSH independientes se unen para formar una única huella digital, denominada en la presente descripción "**F-LSH**". Más específicamente, en estas realizaciones, los componentes del modelador de redes de

objetos, tales como el gestor de objetos y el motor de consultas, se configuran para ejecutar procesos de coincidencia usando varias combinaciones de fragmentos **F-LSH**. Como tal, en estas realizaciones, cada huella digital en conjunto es una "unidad de almacenamiento" eficiente, o vector de bits V_{fp} ancho de W bits. El gestor de objetos genera huellas digitales que caben en un múltiplo par de palabras en la arquitectura de la máquina de destino y, por lo tanto, son compactos y rápidos de manipular con una cantidad mínima de instrucciones de la máquina.

Mientras que el vector interno V_{int} es un vector disperso en un espacio de alta dimensión, V_{fp} es un vector denso en dimensiones de W con subregiones (fragmentos) que se combinan mediante componentes del modelador de redes de objetos para formar claves de indexación. Las ecuaciones 8 y 9 expresan tamaños respectivos de V_{int} y V_{fp} .

Vector de características internas V_{int} :

$$n(V_{int}) \in [1, \infty), \text{espacio} = R^{2^{26}-1}. \quad (8)$$

Huella digital V_{fp} :

$$n(V_{fp}) \equiv W, \quad \text{espacio} = \{0,1\}^W, \quad W \in \{64, 128, 256, 512, \dots\} \quad (9)$$

Los componentes del modelador de redes de objetos se configuran para comparar huellas digitales para determinar las diferencias entre los vectores internos que se representan por las huellas digitales. En particular, aunque estas realizaciones usan la distancia de hamming entre las huellas digitales para optimizar el rendimiento, estas realizaciones no usan la distancia de hamming directamente como una medida de similitud. La distancia de hamming de dos números binarios es el número de posiciones de bits con valores opuestos.

En algunas realizaciones, los componentes del modelador de redes de objetos manifiestan una apreciación de que al menos una propiedad de las huellas digitales F-LSH (como las implementa el modelador de redes de objetos) que hace que las huellas digitales sean adecuadas para la indexación de vectores es que la diferencia moderada entre los vectores internos da como resultado modificación de bits en sus correspondientes huellas digitales. Pequeños cambios en el vector V_{int} dan como resultado cambios pequeños y localizados en la huella digital V_{fp} . Dicho de otra manera, si la diferencia entre V_{int} y V'_{int} es pequeña (unas pocas dimensiones), entonces la diferencia entre V_{fp} y V'_{fp} serán unos pocos bits de valor opuesto dentro de una o más (pero no todas) fragmentos de huellas digitales.

En algunas realizaciones, el gestor de índices usa esta propiedad para crear un conjunto de *índices de permutación* en V_{fp} mezclando y recombinando fragmentos individuales de V_{fp} en subconjuntos más pequeños, denominados en la presente descripción regiones significativas, para usarlos como claves de búsqueda directa en un conjunto de tablas. Como se menciona en la presente descripción, P_j indica un fragmento, R_i indica una región significativa y T_i indica el índice de permutación (tabla de búsqueda) usando R_i como clave.

En algunas realizaciones, el número F de fragmentos V_{fp} en es un parámetro configurable, como lo es el número f_R de fragmentos por región significativa R_i . Todos los R_i pueden contener el mismo número de fragmentos, y el número de bits en R_i puede variar ligeramente ya que no todos los P_j tienen necesariamente el mismo ancho. Esto se ilustra a continuación en la Figura 15 para $F = 5, f_R = 2$. En estas realizaciones, el número de tablas de consulta (N) requeridas para la cobertura completa del índice varía con F y f_R de acuerdo con la Ecuación 10:

$$N = \text{Comb}(F, f_R) = \binom{F}{f_R} = \frac{F!}{f_R!(F-f_R)!} \quad (10)$$

Por ejemplo, una configuración que divide V_{fp} en $F = 4$ fragmentos y usa $f_R = 1$ fragmento por región significativa requiere $N = \text{Comb}(4, 1) = 4$ tablas de búsqueda. Una configuración más grande que divide V_{fp} en $F = 5$ fragmentos y usa $f_R = 2$ fragmentos por región significativa, como se ve arriba, requiere $N = \text{Comb}(5, 2) = 10$ tablas de búsqueda. Como se menciona en la presente descripción, (F, f_R) denota una configuración particular de fragmentos y regiones por fragmento.

Se aprecia que uno de los beneficios de los conjuntos de tablas más grandes es una mayor robustez frente a los cambios en el vector de entrada V_{int} y un mayor rendimiento de búsqueda por tabla. Las desventajas pueden incluir una mayor huella de memoria y mayores demandas de CPU para realizar búsquedas simultáneas en todas las tablas. También se aprecia que siempre que las variaciones de bits entre dos huellas digitales estén aisladas en no más de fragmentos $F-f_R$, existe al menos una región significativa idéntica para las dos huellas digitales, lo que da como resultado una coincidencia en al menos una de las tablas.

Generación de huellas digitales

El gestor de objetos se configura para generar uno o más fragmentos de una huella digital V_{fp} de un vector interno V_{int} mediante la ejecución de un proceso de generación de huellas digitales 4000 ilustrado por las Figuras 16 a la 17 y la 40. El vector interno V_{int} , se muestra en la parte superior de la Figura 16.

Con breve referencia a la Figura 40, se muestra un diagrama de flujo que ilustra un proceso de generación de huellas digitales. El proceso de generación de huellas digitales 4000 se configura para generar huellas digitales que tienen 4 fragmentos, aunque las realizaciones descritas en la presente descripción no se limitan a un número particular de fragmentos. Un ejemplo de huella digital se muestra generalmente en la parte inferior de la Figura 16. Como se muestra en la Figura 40, el proceso de generación de huellas digitales 4000 incluye actos de seleccionar un siguiente elemento sin procesar del vector interno, seleccionar un fragmento objetivo de la huella digital que se está generando, generar un resumen de información de dimensión dentro del elemento seleccionado, ponderar el resumen, combinar el resumen con resúmenes anteriores, determinando si existen elementos adicionales no procesados del vector interno y almacenando el resumen combinado como el fragmento de destino. El proceso de generación de huellas digitales 4000 comienza en 4002.

En el acto 4004, se selecciona un siguiente elemento sin procesar del vector interno. En el acto 4006, se selecciona un fragmento objetivo de la huella digital que se está generando. En una realización, el gestor de objetos selecciona el fragmento de destino analizando la información de dimensión almacenada en el elemento seleccionado. La Figura 16 ilustra el proceso para transformar V_{int} en V_{fp} usando diferentes partes de dim_n como 1) un patrón de bits para la generación de LSH y 2) enrutamiento de fragmentos. Por ejemplo, el gestor de objetos puede usar los 8 bits superiores de la dimensión dim_n para seleccionar un fragmento P_j de V_{fp} que se verá afectado por los bits inferiores b de dim_n . En este ejemplo, la selección de fragmentos se calcula como $j = 1 + Superior_8(dim_n) \bmod F$. El patrón de bits resultante P_j puede calcularse, en un ejemplo, usando una función hash acumulativa que combina los patrones de bits individuales, ponderados por mag_n , de todo el $Inferior_b(dim_n)$.

En el acto 4008 se genera un resumen del elemento seleccionado. En una realización, el gestor de objetos genera el resumen calculando un valor hash de información de dimensión almacenada en el elemento seleccionado. Por ejemplo, el gestor de objetos puede calcular un valor hash para los b bits inferiores de dim_n , donde b es igual al número de bits en el fragmento de destino.

En el acto 4012 se pondera el resumen. En una realización, el gestor de objetos pondera el resumen de acuerdo con la magnitud almacenada en el elemento seleccionado. Por ejemplo, el gestor de objetos puede calcular un factor de ponderación como un porcentaje de una magnitud máxima representada por la magnitud almacenada del elemento seleccionado. En el acto 4014, el resumen ponderado se combina con los resúmenes ponderados anteriores. Los actos 4012 y 4014 se ilustran en la Figura 16. En el acto 4016, se determina si existen elementos adicionales no procesados del vector interno. Si es así, el proceso de generación de la huella digital vuelve al acto 4004. De lo contrario, la generación de huellas digitales almacena los resúmenes ponderados combinados como el fragmento de destino (cualquiera de $P1-P4$ en la Figura 16) en el acto 4018. El proceso de generación de huellas digitales 4000 finaliza en 4020.

La Figura 17 ilustra un ejemplo de las leyes 4004 a 4016. De acuerdo con la Figura 17, un generador de LSH ejecutado por el gestor de objetos determina un patrón de bits final mediante la "votación" bit a bit de los patrones de bits individuales de todas las dimensiones del vector contribuyente, donde el peso de cada "voto" individual es igual a la magnitud del elemento del vector. En el ejemplo ilustrado en el ejemplo Figura 17, tres elementos vectoriales $\{(10010,20), (11001,15), (00011,10)\}$ se combinan para formar el patrón final 10011. También tenga en cuenta que si el tercer elemento del vector fuera 11111 en lugar de 00011, el patrón final sería 11011 en lugar de 10011 (es decir, una diferencia de un solo bit en el valor de LSH como resultado de un elemento de vector diferente).

La Figura 18 resume la transformación del objeto original a través de las tres fases de reducción de dimensionalidad implementadas dentro del modelador de red de objetos 202.

Coincidencia de similitud

El gestor de objetos asocia implícitamente los objetos en función de la similitud de los vectores asociados con los objetos. De acuerdo con diversas realizaciones, la decisión de unir dos objetos en función de la similitud puede configurarse usando una variedad de parámetros configurables que incluyen umbrales dinámicos y una combinación de reglas. Las reglas pueden analizar las características de los objetos potencialmente coincidentes, como sus atributos, su rango, cardinalidad, tamaño del grupo, dispersión del grupo, usando umbrales dinámicos, etc. El gestor de objetos se configura para unir objetos donde se cumple la Desigualdad 11:

$$\text{"los vectores } \mathbf{A} \text{ y } \mathbf{B} \text{ son similares si } Sim(\mathbf{A}, \mathbf{B}) \geq \text{umbral"} \quad (11)$$

Como apreciarán los expertos en la técnica, las técnicas y los sistemas descritos en la presente descripción también permiten esquemas de decisión más complejos, teniendo en cuenta otros atributos de los objetos coincidentes, su rango, cardinalidad, tamaño de grupo, dispersión de grupo, usando dinámica de umbrales, etc.

La Figura 18 es un diagrama de secuencia que ilustra un proceso de reducción de datos en línea con las enseñanzas anteriores. Más específicamente, la Figura 18 ilustra un método ilustrativo para transformar un objeto en varias

representaciones (por ejemplo, uno o más vectores externos, uno o más vectores internos y una huella digital que comprende uno o más fragmentos).

Como motivación para la siguiente discusión, se considera un enfoque ingenuo que usa una base de datos SQL estándar para el almacenamiento y la búsqueda de vectores. En este enfoque, los objetos pueden representarse como filas en una tabla de base de datos y tener una columna para el vector de características original V_{ext} presentado anteriormente. Si se desea recuperar todas las filas en la base de datos cuyos vectores son similares al vector de sondeo Q_{ext} , el primer desafío es el propio lenguaje SQL, que no admite este tipo de consulta directamente. En consecuencia, una solución podría incluir escribir un procedimiento almacenado con las matemáticas necesarias, pero esto requeriría realizar exploraciones de tablas lineales para cada consulta, lo cual es ineficiente. Además, podría ser posible almacenar las dimensiones del vector en columnas o tablas separadas, agregar índices, usar esquemas de combinación elaborados, etc., pero, en última instancia, la tecnología de base de datos existente convencional no está optimizada para funcionar bien para este tipo de aplicación en un gran escala.

En consecuencia, el sistema de base de datos y las técnicas descritas en la presente descripción se diseñan, en parte, para optimizar el tipo de consulta anterior, superando así muchas de las limitaciones mencionadas anteriormente asociadas con los sistemas de base de datos convencionales. Específicamente, el sistema de base de datos y las técnicas descritas en la presente descripción se diseñan para lograr (1) recuperación máxima; (2) máxima precisión; (3) latencia de consulta mínima; y (4) volumen máximo de consultas (rendimiento).

Como se menciona en la presente descripción, la *recuperación* (R) y la *precisión* (P) se definen a través de la noción de *verdaderos positivos* (tp), *falsos positivos* (fp) y *falsos negativos* (fn). La *recuperación* y la *precisión* pueden expresarse respectivamente como las Ecuaciones 12 y 13:

$$\begin{aligned} \text{Precisión} = P &= \frac{tp}{tp+fp} & (12) \\ \text{Recuperación} = R &= \frac{tp}{tp+fn} & (13) \end{aligned}$$

Como se usa en la presente descripción, los verdaderos positivos (tp) son resultados esperados de una consulta que se incluyen dentro del resultado de la consulta. Los falsos positivos (fp) son resultados de consultas que no son resultados esperados. Los falsos negativos (fn) son omisiones incorrectas en los resultados de consulta, es decir, resultados que se esperaban, pero que no se devolvieron. De las definiciones anteriores, se observan dos casos patológicos: 1) una consulta que siempre devuelve *todo* produce un 100% de recuperación ya que ninguna omisión hace $fn=0$, y 2) una consulta que siempre devuelve *nada* produce un 100% de precisión ya que cero resultados hacen $fp=0$ (y consideramos $0/0 \rightarrow 1$). En un ejemplo, los sistemas y técnicas descritos en la presente descripción se diseñan de tal manera que una consulta devuelve algo manteniendo tanto la recuperación como la precisión lo más alto posible. Por lo tanto, en un ejemplo de coincidencia de similitud, "esperado" puede diseñarse de la siguiente manera: si $Sim(Q, V) \geq \tau$ entonces esperamos que V se devuelva de una consulta de Q, de lo contrario no.

El motor de consulta se configura para ejecutar un proceso de consulta que se divide en cuatro etapas. Estas etapas incluyen una *fase de codificación*, una fase de *recuperación*, una *fase de precisión* y una *fase de antisolapamiento*. La fase de codificación primero convierte un objeto de sondeo (Q) en un vector de características Q_{ext} , un vector interno Q_{int} y una huella digital Q_{fp} . La ejecución del motor de consulta de la fase de recuperación se diseña para recopilar tantos **candidatos positivos verdaderos** como sea posible, lo más **rápido** posible, mientras se **minimizan los falsos negativos** y se **limitan los falsos positivos**. La ejecución de la fase de precisión del motor de consulta está diseñada para **eliminar la mayoría de los falsos positivos** producidos durante la fase de recuperación. La ejecución del motor de consultas de la fase de antisolapamiento está diseñada para **eliminar cualquier falso positivo restante** producido durante la fase de precisión.

Se aprecia que puede lograrse un 100 % de recuperación y un 100 % de precisión al devolver todos los vectores en la fase de recuperación, omitiendo la fase de precisión y confiando en la fase de antisolapamiento para filtrar todos los falsos positivos. Sin embargo, este enfoque no proporciona un rendimiento adecuado para redes de objetos de tamaño considerable. Por lo tanto, en algunas realizaciones, el motor de consulta se configura para maximizar la recuperación y la precisión utilizando los procesos de codificación de vectores y huellas digitales descritos anteriormente con referencia a las Figuras 12, 13, 16 y 17, que se aplica como se ilustra en la Figura 18.

La Figura 19 ilustra un proceso de consulta que incluye actos de una fase de codificación 1, una fase de recuperación 2, una fase de precisión 3 y una fase de antisolapamiento 4. Cada una de estas fases se describe más adelante.

En la fase de codificación 1, el motor de consultas prepara, a partir del objeto de sondeo (referido como "Q" en la Figura 19), un vector de características Q_{ext} , un vector interno Q_{int} y una huella digital Q_{fp} para usar durante la comparación y el filtrado. Las técnicas para generar vectores externos, vectores internos y huellas digitales se exponen anteriormente y, por lo tanto, no se describen en detalle en esta sección.

Además, de acuerdo con una realización, en la fase de recuperación 2, el motor de consulta compara la huella digital Q_{fp} con todos los $\{V_{fp}\}$ almacenados usando *índices de permutación* para un alto rendimiento y una prevención

razonablemente buena de falsos negativos. El motor de consulta aplica un umbral de distancia de hamming para un mayor rendimiento y (probable) reducción de falsos positivos. El motor de consultas también puede hacer coincidir el vector Q_{int} interno con todos los $\{V_{int}\}$ almacenados usando tantos índices de dimensión directa como pueda tolerarse (es decir, usando los índices de dimensión directa creados por el gestor de índices) mientras cumple con los requisitos de rendimiento para minimizar el número de falsos negativos. Al final de esta fase, el motor de consulta devuelve un conjunto combinado y desduplicado de candidatos de todos los índices.

Dentro de la fase de precisión 3, el motor de consulta aplica, para todos los candidatos $\{V_{int}\}$ devueltos de la fase de recuperación, el criterio de similitud vectorial $Sim(Q_{int}, \{V_{int}\}) \geq \tau$ para filtrar la *mayoría* de los falsos positivos. Los falsos positivos aún pueden filtrarse debido al solape de dimensión causado, por ejemplo, por la función hash de 26 bits y el "problema de cumpleaños" discutido anteriormente.

Dentro de la fase de antisolapamiento 4, el motor de consulta, para todos los resultados devueltos desde la fase de precisión, recupera los metadatos del objeto $\{V_{ext}\}$ y aplica el criterio de similitud de vectores $Sim(Q_{ext}, \{V_{ext}\}) \geq \tau$ para filtrar todos los resultados falsos positivos devueltos de la fase de precisión. Esta fase puede requerir que V_{ext} se incluya como metadatos "blob" con todos los objetos almacenados y puede requerir mayores demandas de almacenamiento, mayor E/S de disco y mayor latencia de consulta.

La Figura 19 ilustra el funcionamiento de un motor de consulta a un alto nivel. El modelador de redes de objetos admite subprocesos múltiples del motor de consultas, lo que permite que varias consultas se procesen en paralelo usando muchos núcleos de CPU (o GPU) simultáneamente. Algunos componentes admiten una ejecución paralela ilimitada (sin recursos compartidos), mientras que otros requieren trabajo de bloqueo de recursos en subpartes del estilo de "cinta transportadora" de consulta (enfoque canalizado). En algunas realizaciones, este último se aplica a los índices de permutación y dimensión.

Las configuraciones con una mayor cantidad de índices se benefician directamente de las arquitecturas de máquinas de varios núcleos con un alto número de CPU. La ejecución también puede escalarse a varias máquinas, cada una de las cuales aloja un subconjunto del conjunto total de índices de permutación y dimensión. Una configuración de varias máquinas se beneficia al permitir una búsqueda de índice simultánea real en muchos (o todos) los índices configurados para la *misma* consulta, a expensas de requerir un etapa adicional de unir/deduplicar para los candidatos que usan un componente de despachador (no se muestra).

Como se ha descrito anteriormente con referencia a la Figura 19, las realizaciones descritas en la presente descripción se configuran para ejecutar una fase de recuperación durante la ejecución de la consulta. Uno de los propósitos de una fase de recuperación es devolver un conjunto $C = \{V\}$ de coincidencias candidatas para una consulta Q , que cumpla parcialmente los criterios $Sim(Q, V) \geq \tau$ de modo que se minimicen los falsos negativos y se limiten los falsos positivos. El motor de consulta hace referencia a una combinación de índices de permutación e índices de dimensión para lograr este propósito. Una condición de coincidencia positiva se define en términos de umbral de similitud de modo que: clase positiva = $Sim(Q, V) \geq \tau$ y clase negativa = $Sim(Q, V) < \tau$.

La Figura 20 ilustra gráficamente los candidatos producidos en la fase de recuperación. La naturaleza del mecanismo de recuperación del motor de consultas es "lanzar una red", lo que significa que el cuadro de candidato de recuperación en la figura se extiende tanto hacia la derecha como hacia la izquierda en un esfuerzo por minimizar los falsos negativos. Sin embargo, la extensión de la izquierda al territorio de falsos positivos siempre es mucho más pequeña que el conjunto de verdaderos negativos, lo que garantiza un conjunto manejable de resultados candidatos para la fase de precisión posterior.

En algunas realizaciones, el motor de consultas se configura para implementar una búsqueda de objetos rápida y altamente escalable por huella digital durante la fase de recuperación de consultas a través de los índices de permutación, como se ilustra en la Figura 21. Como se describió anteriormente en la sección Huellas digitales, para una configuración determinada (F, f_R) , el gestor de objetos genera una huella digital que se divide en fragmentos F que se mezclan y se vuelven a combinar para crear claves de indexación de fragmentos f_R cada uno. Hay N formas diferentes de combinar fragmentos F en conjuntos de fragmentos f_R cada uno, como se expresa por $N = Comb(F, f_R)$. Para la i -ésima combinación, la clave que se forma por un subconjunto seleccionado de fragmentos se denomina en la presente descripción región significativa R_i , donde $i \in \{1, 2, 3, \dots, N\}$. En algunas realizaciones, R_i es la clave para el índice de permutación T_i donde el conjunto completo de índices de permutación se indica como $T = (T_1, T_2, \dots, T_N)$.

En algunas realizaciones, para una consulta de similitud, el motor de consulta se configura para recopilar objetos candidatos durante la fase de recuperación para el índice de permutación T_i como se ilustra en las Figuras 22 y 23. En estas realizaciones, el índice de permutación T_i es una estructura de tipo hash grande, pero simple, con bloqueo a nivel de fila, que permite lecturas y escrituras simultáneas con una probabilidad mínima de contención. Todos los índices de permutación existen en la memoria (RAM) en tiempo de ejecución.

Como se ilustra en la Figura 42, un proceso de recuperación 4200 incluye actos de formar una región siguiente de una huella digital de sondeo, seleccionar una matriz de ranuras de índice correspondiente a la región formada, escanear huellas digitales de objetos incluidas en la matriz de ranuras de índice, determinar si existen regiones adicionales sin

procesar de la huella digital de sondeo, y los candidatos de recuperación que regresan. El proceso de recuperación 4200 comienza en 4202.

En el acto 4204, se forma una región siguiente de una huella digital de sondeo. La Figura 22 se refiere a la huella digital de sondeo Q_{fp} como y se refiere a la región formada como la combinación de P_1 y P_4 . En algunas realizaciones, el motor de consulta forma una región significativa R_i a partir de un subconjunto de bits de la huella Q_{fp} del objeto de sondeo. El número de bits d en R_i dicta la longitud requerida 2^d del índice de permutación T_i . Cada *línea de índice* en T_i es una estructura que contiene un puntero a una *matriz de ranuras de índice de longitud variable*, además de varios bloqueos y datos de contabilidad. La clave de indexación R_i identifica directamente una línea de índice en T_i . Cada ranura en la matriz de ranuras de índice contiene un puntero a un objeto y la huella digital V_{fp} de ese objeto. Todos los objetos en la matriz de ranuras de índice tienen huellas digitales V_{fp} con una región significativa R_i idéntica.

En el acto 4206, se selecciona una línea de índice en base a la línea de índice identificada por la región formada en el acto 4204. Este acto se ilustra en la Figura 22 En una realización, el motor de consultas selecciona la línea de índice identificada.

En la acción 4208, se explora una matriz de ranuras de índice identificada por un puntero en la línea de índice seleccionada. Este acto también se ilustra en la Figura 22 En una realización, el motor de consulta explora la matriz de ranuras de índice identificada por el puntero en la línea de índice seleccionada. La matriz de ranuras de índice puede explorarse ranura por ranura de principio a fin seleccionando ranuras cuya huella digital difiere como máximo en h posiciones de bits, es decir, $\text{distancia_hamming}(Q_{fp}, v_{fp}) \leq h$. El motor de consulta agrega cada objeto con la huella digital V_{fp} que está dentro de la distancia de hamming configurada h de Q_{fp} a una matriz de candidatos de recuperación, formando así el resultado de la consulta de la fase de recuperación del índice de permutación. En al menos una realización, los objetos se agregan como punteros dentro de la matriz de candidatos de recuperación. Este acto también se ilustra en la Figura 22.

En el acto 4210, se determina si quedan regiones adicionales sin procesar de la huella digital de sondeo. Si es así, el proceso de recuperación 4200 vuelve al acto 4204. De lo contrario, el proceso de recuperación 4200 devuelve la matriz de candidatos de recuperación en el acto 4212. El proceso de recuperación finaliza en 4214.

Como se muestra en la Figura 22, en algunas realizaciones, los propios objetos residen en el *almacén de objetos*. En al menos una realización, solo hay un almacén de objetos por instancia de modelador de red de objetos (máquina). De acuerdo con esta realización, cuando el modelador de red de objetos se implementa en varias máquinas, hay un almacén de objetos por máquina. Todos los índices $T \subseteq T$ por máquina comparten el mismo almacén de objetos al hacer referencia a objetos a través de punteros.

En algunas realizaciones, el gestor de índices mantiene una copia de la huella digital V_{fp} en cada ranura de índice (V_{fp} también existe en el objeto) para eficiencia computacional. En estas realizaciones, una matriz de ranuras de índice es una pieza de memoria contigua que puede recuperarse de la memoria principal solo una vez y puede caber en la memoria caché L1 durante la exploración lineal, evitando así la desreferenciación de cada puntero de objeto para acceder a la huella digital del objeto.

En algunas realizaciones, la exploración hacia abajo de una matriz de ranuras de índice es muy eficiente y requiere solo unas pocas instrucciones en un bucle cerrado que accede solo a la memoria L1 rápida y a los registros. En estas realizaciones, el motor de consulta ejecuta el cálculo de la distancia de hamming en dos instrucciones únicamente: XOR bit a bit seguido de POPCNT intrínseco, que forma parte de la microarquitectura x86. Cada vez que el motor de consultas encuentra una ranura de índice que tiene V_{fp} dentro de la distancia de hamming h de Q_{fp} , el puntero de objeto de la ranura se agrega al resultado de la consulta de la fase de recuperación.

En algunas realizaciones, el motor de índice se configura para insertar un nuevo objeto en el índice de permutación T_i escaneando hacia abajo la matriz de ranuras de índice hasta que se encuentra una ranura vacía ($V_{fp} = 0$), o se alcanza el final de la matriz en cuyo punto la matriz de ranuras de índice se expande. En estas realizaciones, el motor de indexación se configura para eliminar un objeto del índice de permutación T_i estableciendo el puntero del objeto en NULO y estableciendo $V_{fp} = 0$. Además, en estas realizaciones, el motor de indexación se configura para reducir y desfragmentar una matriz de ranuras de indexación cuando la utilización cae por debajo de un cierto umbral.

En algunas realizaciones, el gestor de índices ejecuta asignadores especializados que se han implementado para admitir la expansión y reducción constante y dinámica de matrices de ranuras de índice con máxima eficiencia y mínima cantidad de fragmentación de memoria. En estas realizaciones, el índice de permutación T_i está estructurado de acuerdo con la ilustración simplificada de la Figura 23, con matrices de ranuras de índice de tamaño variable que residen fuera de la tabla principal. El gestor de índices intercambia una matriz de ranuras de índice existente por una matriz más grande o más pequeña extrayendo una nueva matriz del asignador más grande o más pequeño, copiando huellas digitales y punteros de objetos de la matriz existente a la nueva, desfragmentando según sea necesario y luego proporcionando la matriz antigua de vuelta a su asignador para su reciclaje.

En algunas realizaciones, el motor de consultas se configura para implementar una búsqueda de objetos rápida y altamente escalable por índice de dimensión directa durante la fase de recuperación de consulta a través de los índices de dimensión, como se ilustra en la Figura 24. En algunas realizaciones, el índice de dimensión directa se implementa usando la misma estructura de índice que el índice de permutación. Sin embargo, en lugar de procesar varias dimensiones vectoriales y convertirlas en un fragmento de huella digital como se hizo para el índice de permutación, el índice de dimensión utiliza la dimensión de vector individual dim_n como clave de indexación y búsqueda.

Como se menciona en la presente descripción, el conjunto total de índices de dimensión puede expresarse como $D = \{D_1, D_2, \dots, D_N\}$. En algunas realizaciones, el motor de consultas identifica V_{int} usando la dimensión dim_n . Una configuración determinada del modelador de redes de objetos puede incluir cero o más índices de dimensión según las necesidades de rendimiento y el nivel de recuperación garantizado requerido.

En general, cuando el gestor de índices utiliza M índices de dimensión directa, el gestor de índices indexa dim_n , $n \in \{1 \dots M\}$ en $D_i = n \in \{1 \dots M\}$. Durante la fase de recuperación de una consulta, cuando el motor de consulta compara Q_{int} con V_{int} , el motor de consulta compara las M primeras dimensiones de Q_{int} con las M primeras dimensiones de V_{int} , lo que significa que, $dim_{n'} \in Q_{int}$ $n \in \{1 \dots M\}$ se comparan con $dim_{n'} \in V_{int}$ $n \in \{1 \dots M\}$ que requieren operaciones de búsqueda M^2 . Debido a que el tiempo de ejecución de consultas para índices de dimensión directa crece cuadráticamente con M , es beneficioso mantener M lo más bajo posible.

Para una aplicación que requiere un alto nivel de recuperación, el gestor de índices puede configurar M tan alto como sea necesario, hasta el número máximo de dimensiones vectoriales en uso para V_{int} para una recuperación del 100 %. En algunas realizaciones, por lo tanto, es posible ampliar el cuadro de candidatos de recuperación que se ilustra en la Figura 20 completamente a la derecha, eliminando todos los falsos negativos, aunque a costa de rendimiento.

Se aprecia que, para una aplicación dada a , existe un umbral ρ_a , de modo que $M > \rho_a$ produce suficiente recuperación. Al mismo tiempo, existe un umbral τ_a , de modo que $M < \tau_a$ produce un rendimiento suficiente. El modelador de redes de objetos resuelve la maldición de la dimensionalidad para la aplicación a cuando $\rho_a < M < \tau_a$. Para satisfacer $\rho_a < M < \tau_a$ puede ser necesario implementar el modelador de redes de objetos en varias máquinas donde cada máquina sirve un subconjunto $D' \subseteq D$.

En algunas realizaciones, el motor de consulta se configura para fusionar candidatos de recuperación identificados a través de los índices de permutación y dimensión directa, como se ilustra en la Figura 25. Todos los índices de permutación $T_i \in T$ e índices de dimensión $D_i \in D$ proporcionan indexación independiente y búsqueda de todos los objetos en el almacén de objetos. Como tal, un objeto A será indexado por diferentes "vistas" (regiones significativas) de V_{fp} en T y por diferentes dimensiones de V_{int} en D . Dado que el motor de consultas repite una consulta Q para todos los T y D , es completamente posible para que el objeto A se encuentre en varios (o en todos) los índices. Por lo tanto, la fase de recuperación no se completa hasta que el motor de consultas fusiona y elimina los duplicados de los conjuntos individuales de candidatos de todos los índices, de modo que cualquier objeto candidato aparece exactamente una vez en el conjunto final de candidatos.

En algunas realizaciones, para ejecutar este proceso, el motor de consulta se configura para atravesar y colocar todos los resultados candidatos de todos los índices en un hash de eliminación de duplicados. Varios hashes de deduplicación asignados estáticamente están disponibles para permitir la ejecución de subprocesos múltiples, y los hashes existen en diferentes tamaños para acomodar cualquier conjunto de resultados de la manera más eficiente desde el punto de vista computacional. Estas realizaciones mantienen la mayor parte de la actividad en las cachés de CPU más internas para el subproceso actual mientras minimizan la invalidación de caché para otros subprocesos. Además, en estas realizaciones, el motor de consultas devuelve un conjunto de candidatos fusionados y deduplicados como resultado final de la fase de recuperación de consultas.

Como se ha descrito anteriormente con referencia a la Figura 19, las realizaciones descritas en la presente descripción se configuran para ejecutar una fase de precisión durante la ejecución de la consulta. En una realización, el motor de consultas ejecuta la fase de precisión. En esta realización, después de que la fase de recuperación haya reunido tantas coincidencias de similitud probables como sea posible, la fase de precisión filtra todos los falsos positivos, excepto aquellos enmascarados por el solape de dimensión en V_{int} . En algunas realizaciones, aunque el motor de consulta identifica y elimina los falsos positivos calculando la Ecuación 11 para el objeto de sondeo y cada objeto que permanece en el conjunto de candidatos de recuperación. La fase de precisión, de acuerdo con una realización, se ilustra en la Figura 26.

Como se ha descrito anteriormente con referencia a la Figura 19, las realizaciones descritas en la presente descripción se configuran para ejecutar una fase de antisolapamiento durante la ejecución de la consulta. En una realización, el motor de consultas ejecuta la fase de antisolapamiento. Se aprecia que, debido al solape de dimensión vectorial en V_{int} como resultado de la codificación de características de 26 bits, es posible que se enmascaren falsos positivos durante la fase de precisión. Es decir, $Sim(Q_{int}, V_{int}) \geq \tau$ no implica $Sim(Q_{ext}, V_{ext}) \geq \tau$. En general, el solape de dimensión puede expresarse como la Ecuación 14:

$$\delta = Sim(Q_{int}, V_{int}) - Sim(Q_{ext}, V_{ext}), \quad \tau \gg \delta > 0 \quad (14)$$

Aunque δ suele ser pequeño y, a menudo, cero para las comparaciones de vectores, el motor de consultas puede configurarse para aplicar la fase de antisolapamiento antes de devolver el resultado final de la consulta a una aplicación que requiere una precisión absoluta. El antisolapamiento se realiza aplicando el filtro $\text{Sim}(Q_{ext}, V_{ext}) \geq \tau$ al resultado producido en la fase de precisión, como se ilustra en la Figura 27. En algunas realizaciones, el motor de consulta requiere que V_{ext} se incluya como metadatos de objeto para habilitar el antisolapamiento.

Resumen formalizado de la ejecución de la consulta

Para resumir formalmente, la ejecución de consultas de acuerdo con al menos una realización sea V el conjunto completo de vectores indexados y sea C el conjunto de candidatos producidos en la fase de recuperación usando índices de permutación e índices de dimensión. Además, sea S_τ el conjunto real de vectores cuya similitud con la consulta Q_{int} es mayor o igual que τ , es decir, la clase positiva, como se describe en la Ecuación 15:

$$S_\tau = \{V_{int} \in V | \text{Sim}(Q_{int}, V_{int}) \geq \tau\} \quad (15)$$

Para un índice de permutación dado $T = \{T_1, T_2, \dots, T_N\}$ configurado como (F, f_R) existe un umbral mínimo de similitud y de modo que cuando se aplica en el contexto de la consulta Q_{int} no dará como resultado en cualquier falso negativo. Las diferencias entre cualquier V_{int} y Q_{int} en el conjunto positivo S_τ son lo suficientemente pequeñas como para garantizar que al menos una región significativa R_i quede intacta entre cualquier V_{fp} y Q_{fp} .

$$\exists \gamma : C \supseteq S_\gamma \quad (16)$$

También existe un umbral de similitud más pequeño $\gamma - \delta$ de modo que cuando se aplica en el contexto de la consulta Q_{int} siempre dará como resultado falsos negativos. Las diferencias entre algunos V_{int} y Q_{int} en el conjunto positivo expandido $S_{\gamma-\delta}$ ahora son lo suficientemente grandes como para causar alteraciones en todas las regiones significativas $R_{i \in (0 \dots N)}$ entre V_{fp} y Q_{fp} , lo que hace que V_{fp} sea irrecuperable en cualquier tabla T_n .

$$\exists \delta : S_{\gamma-\delta} \supset S_\gamma \Rightarrow C \not\supseteq S_{\gamma-\delta} \quad (17)$$

Estos conceptos se ilustran en la Figura 28. Cuando establecemos nuestra expectativa de que la consulta Q_{int} devuelva vectores con similitud $\tau \geq \gamma$, podemos capturar todo el conjunto S_γ dentro de los candidatos de recuperación C . Diferentes consultas tienen características de recuperación ligeramente diferentes para la misma configuración, indicadas por la "variabilidad C " entre los límites sólido y punteado de C , que es capaz de moverse horizontalmente mientras permanece dentro de un "potencial de recuperación".

Cuando elevamos nuestras expectativas bajando el umbral de similitud a $\tau < \gamma$, vemos que partes de la clase positiva (segundo círculo concéntrico sólido desde el centro) pueden caer dentro de C y otras partes de la clase positiva pueden caer fuera para grados variables de C dentro del potencial de recuperación. Los falsos negativos ocurren de vez en cuando con una probabilidad ligeramente superior a cero cuando τ cae justo por debajo de γ , y una probabilidad ligeramente inferior a uno cuando $\tau \rightarrow \gamma - \delta$.

A medida que aumentamos aún más nuestra expectativa de recuperación al reducir el umbral de similitud a $\tau < \gamma - \delta$, notamos que $S_{\gamma-\delta}$ marca el límite para la clase positiva en la que son posibles los resultados de la consulta sin falsos negativos. Yendo más allá de $S_{\gamma-\delta}$ como con $\tau < \gamma - \delta$, nos damos cuenta de que todos los resultados de las consultas sufrirán falsos negativos.

El valor de δ depende de la naturaleza de los vectores indexados V y normalmente varía de una consulta a otra. Sea $P(fn|Q_{int}, V)$ la probabilidad de falsos negativos dada la consulta Q_{int} contra un conjunto particular de vectores V . Entonces $\tau \geq \gamma$ garantiza cero falsos negativos y $\tau < \gamma - \delta$ garantiza uno o más falsos negativos. En el espacio entre γ y $\gamma - \delta$ es posible, pero no seguro, que una consulta sufra falsos negativos.

$$P(fn|Q_{int}, V) = \begin{cases} 0, & \tau \geq \gamma \\ \lambda(\tau), & \gamma > \tau > \gamma - \delta \\ 1, & \tau < \gamma - \delta \end{cases} \quad (18)$$

El valor de la función $\lambda(\tau)$ es cercano a 0 para $\tau \approx \gamma$ y cercano a 1 para $\tau \approx \gamma - \delta$.

Como ejemplo, haciendo referencia de nuevo a la Figura 28, si $\gamma = 0,9$ y $\delta = 0,7$, entonces una consulta Q_{int} contra V nunca devolverá falsos negativos siempre que solicitemos coincidencias con una similitud de 0,9 o superior, es decir, $\tau \geq 0,9$. (El círculo más interno siempre está contenido en C .) Si ampliamos nuestros criterios de coincidencia y solicitamos coincidencias con $\tau = 0,0$, es posible que no tengamos una recuperación del 100 %, es decir, es posible que se produzcan falsos negativos. Una consulta aún más amplia con $\tau = 0,7$ tiene una mayor probabilidad de falsos negativos, y así sucesivamente. (Ahora estamos en territorio $\lambda(r)$ fuera del círculo más interno.) Finalmente, una consulta con un umbral de similitud $\tau = 0,9 - 0,7 = 0,2$ o menor siempre tendrá falsos negativos.

Para cumplir con el objetivo principal de minimizar los falsos negativos en la fase de recuperación, debemos minimizar γ , maximizar δ y hacer que $\lambda(r)$ sea lo más convexo posible (con tendencia a 1 lo más tarde posible). En la práctica, puede ser difícil cuantificar γ , δ y $\lambda(r)$ analíticamente. Debido a su dependencia de la consulta Q_{int} y la naturaleza de los vectores indexados $\mathbf{V}$, no tiene sentido definirlos en términos de valores absolutos. Sin embargo, permiten que se hagan ciertas afirmaciones cualitativas con respecto al uso de la indexación de permutaciones y la indexación de dimensiones para lograr un objetivo general.

Podemos controlar γ directamente y δ indirectamente a través de la elección de (F, f_R) para el índice de permutación $\mathbf{T}$. En general, aumentando el número de fragmentos F en la huella digital V_{fp} , hacemos que cada fragmento sea más pequeño y, por lo tanto, menos probable que se vea afectado por la alteración de bits de F-LSH. Al mismo tiempo, al aumentar el número de fragmentos f_R por región significativa, aseguramos

R_i contiene suficientes bits para una búsqueda eficiente (es decir, se minimiza la longitud de la matriz de ranura de índice promedio que requiere un escaneo lineal) Como resultado de hacer que V_{fp} sea más robusto, reducimos γ , es decir, podemos especificar un umbral de similitud más pequeño sin ejecutar un riesgo de falsos negativos. Sin embargo, aunque δ y $\lambda(r)$ se mejoran indirectamente, no son el objetivo principal cuando se expande el índice de permutación.

En cambio, podemos controlar δ y $\lambda(r)$ expandiendo el índice de dimensión $\mathbf{D} = \{D_1, D_2, \dots, D_N\}$. En general, al indexar más dimensiones de V_{int} , aseguramos que una porción más grande de $\mathbf{S}$ pueda encontrarse mediante la consulta Q_{int} , lo que aumenta δ , es decir, las porciones de $\mathbf{S}$ que están enmascaradas en el índice de permutación $\mathbf{T}$ debido a la alteración de bits en todas las regiones significativas ahora se vuelven visibles ya que las dimensiones individuales del vector se indexan directamente. Al asegurarnos de que las dimensiones más ponderadas sean las indexadas en $\mathbf{D}$ (y consultadas), hacemos que $\lambda(r)$ sea máximamente convexa. Claramente, expandir el índice de dimensión también afecta positivamente a γ , pero generalmente a un costo más alto que usar el índice de permutación. Por lo tanto, el propósito principal del índice de dimensión es controlar δ y $\lambda(r)$.

También notamos que el índice de permutación $\mathbf{T}$ puede crecer mucho debido a que $Comb(F, f_R)$ requiere más recursos de memoria y computación paralela para lograr un γ más bajo con un alto rendimiento de consulta. Las mismas características de recuperación también pueden lograrse a través del índice de dimensión $\mathbf{D}$ solo a un menor costo de recursos (hardware), pero con un rendimiento de consulta inferior. En el extremo, podemos implementar el índice de dimensión $\mathbf{D}$ solo, con el número de dimensiones indexadas M igual al número de dimensiones en V_{int} . Esta configuración obviamente tiene características de recuperación superiores ($\gamma \rightarrow 0$), pero también es extremadamente ineficiente a menos que el número de dimensiones configuradas para V_{int} sea pequeño. (Para sistemas con V_{int} pequeños, indexar exclusivamente a través de $\mathbf{D}$ puede ser la solución óptima).

Elegir una configuración adecuada es una cuestión de equilibrar el tamaño N de $\mathbf{T}$ a través de (F, f_R) con el tamaño de $\mathbf{D}$ a través de M . Se debe hacer una elección informada teniendo en cuenta la latencia de consulta máxima aceptable y los costos de hardware, el impacto general de falsos negativos en la aplicación, la probabilidad aceptable de falsos negativos y, lo que es más importante, el grado de coincidencia aproximada requerida. Esto último es importante porque si una aplicación de coincidencia aproximada necesita operar en, por ejemplo, $\tau \geq 0,9$ con recuperación garantizada, el costo de lograr $\gamma = 0,9$ es mucho menor que el de lograr, digamos, $\gamma = 0,5$. Además, si la aplicación puede operar adecuadamente sin recuperación garantizada, la implementación es aún más simple.

Lograr una alta precisión es cuestión de aplicar la función de filtro adecuada a $\mathbf{C}$. Dado que tenemos acceso a V_{int} , como parte de los objetos devueltos en $\mathbf{C}$ podemos aplicar $Sim(Q_{int}, V_{int})$ directamente a $\mathbf{C}$ para obtener la alta precisión resultado $\mathbf{R}_a$.

$$\mathbf{R}_a = \{V_{int} \in \mathbf{C} | Sim(Q_{int}, V_{int}) \geq \tau\} \quad (19)$$

En general, esperamos que $\mathbf{R}_a$ contenga falsos positivos con una probabilidad distinta de cero, pero muy pequeña, debido al solape de características al codificar V_{int} desde V_{ext} . Para obtener el resultado $\mathbf{R}$ sin falsos positivos bajo ninguna circunstancia aplicamos otra función de filtro a $\mathbf{R}_a$

$$\mathbf{R} = \{V_{ext} \in \mathbf{R}_a | Sim(Q_{ext}, V_{ext}) \geq \tau\} \quad (20)$$

Tenga en cuenta que la extracción de V_{ext} del resultado con alias $\mathbf{R}_a$ implica un mecanismo de búsqueda secundario que usa el meta almacenamiento. La extracción de V_{ext} de $\mathbf{R}_a$ requiere muchos recursos porque V_{ext} debe recuperarse del disco, mientras que V_{int} está disponible ya que $\mathbf{V}$ reside en la memoria. Como tal, es probable que recuperar V_{ext} sea mucho más costoso que la aplicación real del filtro $Sim(Q_{ext}, V_{ext}) \geq \tau$.

Descripción general de la arquitectura y estructuras de datos

El modelador de redes de objetos descrito en la presente descripción hace un uso intensivo de la memoria para la asociación de objetos y la comparación de similitudes. Para lograr un alto rendimiento, todos los índices pueden existir completamente en la memoria como tablas híbridas de búsqueda hash/lineal, con vaciado de disco bajo demanda para acelerar la inicialización en el futuro arranque del sistema. Todos los componentes del modelador de redes de objetos se crean asumiendo, en un ejemplo, una plataforma de destino de 64 bits y capacidades de servidor de nivel empresarial. (La cantidad de memoria requerida también está disponible rápidamente en hardware de nivel de consumidor).

En algunas realizaciones, el almacén de objetos 214 almacena información descriptiva de objetos dentro de una red de objetos. Los ejemplos de esta información incluyen atributos de objetos, vectores y metadatos. En estas realizaciones, el almacén de objetos 214 reside en la memoria usando la duplicación de disco para mayor durabilidad. En otras realizaciones, el almacén de objetos 214 reside tanto en la memoria volátil como en la no volátil y, por lo tanto, no está limitado a un tipo particular de memoria. Por ejemplo, el componente de meta almacén puede residir solo en el disco. En algunas realizaciones, el modelador de red de objetos 202 se configura para sincronizar el almacenamiento de objetos en memoria 214 y su instantánea basada en disco usando técnicas de escritura simultánea de datos. En estas realizaciones, el almacenamiento de objetos 214 está optimizado para el rendimiento usando registros de compromiso y almacenamiento en caché de disco de flujo continuo de lotes de operaciones.

De acuerdo con algunas realizaciones, un despliegue mínimo del modelador de redes de objetos incluye el almacén de objetos 214, el índice de objetos y el índice inverso. Cada uno de estos componentes se describe en detalle a continuación.

La Figura 29 ilustra los componentes de datos que se utilizan por un modelador de redes de objetos, como el modelador de redes de objetos 202 descrito anteriormente con referencia a la Figura 2, de acuerdo con algunas realizaciones. Como se muestra, los componentes de datos en la Figura 29 incluyen un índice de objeto, un índice inverso, uno o más índices de permutación, uno o más índices de dimensión, un almacén de objetos, una pluralidad de asignadores de matrices de ranuras de índice y una pluralidad de asignadores de vectores. El almacén de objetos incluye uno o más objetos y un meta almacén. Cada uno de los uno o más objetos incluye uno o más vectores. Cada uno de los uno o más vectores incluye una o más características. En algunas realizaciones, el índice de objeto, el índice inverso, uno o más índices de permutación y uno o más índices de dimensión corresponden al almacén de índices 216 descrito anteriormente con referencia a la Figura 2.

Almacén de índices

Como se indicó anteriormente, en algunas realizaciones, el modelador de redes de objetos se configura para hacer un uso intensivo de la memoria para la asociación de objetos y la comparación de similitudes. Por ejemplo, en estas realizaciones, el modelador de redes de objetos logra un alto rendimiento al asignar todos los índices en la memoria como tablas híbridas de búsqueda hash/lineal, con vaciado de disco bajo demanda para acelerar la inicialización al inicio. En algunas realizaciones, todos los componentes del modelador de redes de objetos se configuran para una plataforma de destino de 64 bits y capacidades de servidor de nivel empresarial, aunque las realizaciones no se limitan a una plataforma o servidor de destino en particular.

En estas realizaciones, tanto el índice de objeto 200 como el índice inverso 202 se basan en la memoria. No es necesario configurar una implementación del modelador de redes de objetos para la comparación de similitudes, en cuyo caso la característica de asociación implícita de objetos no está disponible. Sin embargo, la mayoría de los escenarios de implementación incluirán alguna forma de coincidencia de similitud usando uno o más índices de permutación 204 o uno o más índices de dimensión directa 206.

En algunas realizaciones, el modelador de redes de objetos, y más particularmente un gestor de índices, como el gestor de índices 212 descrito anteriormente con referencia a la Figura 2, se configura para implementar un almacén de índices, tal como el almacén de índices 216 descrito anteriormente con referencia a la Figura 2, usando la estructura de datos que se muestra en la Figura 30. En estas realizaciones, cada índice en el almacén de índices usa mecanismos de búsqueda e indexación similares. Además, en estas realizaciones, la estructura linerail forma la base de cada índice en el almacén de índices. La Tabla 2 enumera los atributos de la estructura linerail de acuerdo con algunas realizaciones:

TABLA 2

TIPO	Nombre	Descripción
Int	id	Identificador de tabla 0,1,2,... usado para conjuntos de tablas (perm. y dim.)
simindex_mode_t	mode	Enumeración de modo: permutación (1), dimensión (2), objeto (4), inversa (8)
perm_t	perm	Descriptor de permutación. Define la clave de búsqueda y la longitud de la tabla.
CS_LOCK	vlock	Mutex de variable de estado. Breve bloqueo de tabla completa para adquirir bloqueo de línea de índice.
CS_LOCK	wbusy	Mutex de escritura exclusivo para semáforo singleton por línea de índice
CS_COND	rzero	Transmite lectores de línea de índice cero para permitir que continúe la solicitud de escritura
Int	wbusycnt	Número actual de subprocesos escribiendo o esperando para escribir la línea de índice (0 o 1)
long long	ndirty	Número de operaciones de tabla desde el último vaciado al disco
Long	nrows	Longitud de la tabla, depende de <i>perm</i> .
long long	nslots	Número total de ranuras de índice asignadas para esta tabla
long long	nobj	Número total de objetos indexados en esta tabla
indexline_t*	base	El hash de línea de índice (la tabla)
size_t [64]	size_hist	Histograma del recuento de ranuras en cada línea de índice ($\log_2$ cubos)

En estas realizaciones, la estructura linerail proporciona identificación de tabla, modo tabla, contabilidad y soporte de bloqueo. Además, en estas realizaciones, *el atributo perm* define cómo el modelador de redes de objetos forma una clave de búsqueda de tabla a partir de un *identificador* de búsqueda, que depende del *modo tabla*.

Para un índice de permutación, el identificador es la huella digital: $\text{identificador} = V_{fp}$
 Para un índice de dimensión D_n , el identificador es la dimensión: $\text{identificador} = \text{dim}_n \text{ de } V_{int}$
 Para el índice de objetos, el identificador es la parte alta de OBID: $\text{identificador} = \text{high64}(\text{OBID})$

En algunas realizaciones, el descriptor de permutación, que se define en la Tabla 3 a continuación, especifica cómo los componentes del modelador de redes de objetos que interactúan con los índices, como el gestor de índices o el motor de consultas, identifican los bits dentro de un identificador y los vuelven a ensamblar en una asignación clave de índice a uno de los registros de índice. Para los índices de permutación, los componentes del modelador de redes de objetos reorganizan fragmentos de V_{fp} en una clave de índice. Para otros índices, los componentes del modelador de redes de objetos aplican una máscara de bits a sus identificadores sin mezclar para restringir el tamaño de los identificadores para que coincidan con las *nrows* de la tabla. Por ejemplo, una tabla configurada con *nrows* = 2^{22} = 4 194 304 utilizará una máscara de bits de 22 LSB = 1 para formar la clave de índice del identificador.

TABLA 3

Tipo	Nombre	Descripción
int	nsegm	Número de segmentos (o fragmentos) para dividir el identificador en la reorganización
int	nsign	Número de regiones significativas, es decir, fragmentos de identificador para usar para la clave de índice
int	pno	Número de partición de permutación 0,1,2,... usado por la función de permutación
FP_t*	máscara	Lista de máscaras de bits utilizadas por la función de permutación, tamaño = nsegm
FP_t*	cambio	Lista de cambios de bits usados por la función de permutación, tamaño = nsegm

En algunas realizaciones, la *base* de la tabla de consulta, que se define en la Tabla 4 a continuación, es una matriz de estructuras de *líneas de índice* de longitud *nrows*. En estas realizaciones, los componentes del modelador de red de objetos seleccionan la línea de índice para operar en la que se indica mediante la clave de índice. Además, en estas realizaciones, el gestor de índices mantiene los atributos de contabilidad relacionados con el conjunto completo de la línea de índice en la estructura de rieles lineales y mantiene los atributos contenidos en la estructura de la línea de índice, que rigen el uso de una matriz de ranuras de índices individuales. Así, en estas realizaciones, el atributo *nobj* del linerail es, por lo tanto, la suma de los atributos *nobj* en todas las líneas de índice, y el atributo *nslots* del linerail es la suma de los atributos *sz* en todas las líneas de índice.

TABLA 4

Tipo	Nombre	Descripción
Long	sz	Longitud de la matriz de ranuras de índice
Long	nobj	Número de objetos activos en la matriz de ranuras de índice
Long	wdx	Escriba el índice para la matriz de ranuras de índice (el siguiente objeto se escribirá en esta posición)
Long	rdx	Lea el índice para la matriz de ranuras de índice (las operaciones de búsqueda comienzan en esta posición)
Struct (uin32_t bit field)	res(8)	* reservado *
	wreq(8)	Semáforo singleton (0 o 1), número de subprocesos que escriben en la matriz indexslot
	rcnt(16)	Número de subprocesos que actualmente escanean (leen) la matriz de ranuras de índice
indexslot t*	arr	Conjunto de ranuras de índice que contienen punteros de objetos e identificadores de objetos

En algunas realizaciones, los componentes del modelador de redes de objetos (por ejemplo, subprocesos que ejecutan aspectos del modelador de redes de objetos, como el gestor de índices o el motor de consultas) se configuran para implementar el bloqueo de tablas para permitir el máximo paralelismo en un entorno de ejecución multinúcleo. Por lo tanto, en estas realizaciones, se permite que múltiples hilos accedan al mismo lineraíl simultáneamente. Para evitar el uso excesivo de recursos, en algunas realizaciones, las instancias de exclusión mutua y condición se comparten para todo el lineraíl, mientras que las líneas de índice individuales contienen información sobre la actividad particular actualmente en curso para esa línea de índice. De acuerdo con algunas realizaciones, en cualquier momento dado, las líneas de índice pueden leerse o escribirse, pero no ambas cosas. Además, en algunas realizaciones, varios subprocesos pueden leer la misma línea de índice simultáneamente, mientras que solo un único subproceso puede escribir en una línea de índice.

En algunas realizaciones, los componentes del modelador de red de objetos están configurados para permitir que las operaciones de lectura en la matriz de ranuras de índice *arr* continúen siempre que ningún otro subproceso esté escribiendo en *arr* o esperando para escribir en *arr*. Si es necesario, los componentes del modelador de redes de objetos retrasan una operación de lectura hasta que se completa una operación de escritura simultánea al realizar primero una verificación rápida de *wreq*, y si *wreq* no es cero, bloquear en el bloqueo *wbusy* ahora bloqueado. Normalmente, *wreq* es cero y no se necesita bloquear en *wbusy*. El acceso atómico a todas las variables de estado, incluido el bloqueo *wbusy* secundario, se garantiza a través de *vlock*. Cuando se permite que continúe una operación de lectura, el *rcnt* de la línea de índice se incrementa en uno. Se permiten múltiples lectores, hasta un máximo de 65 535.

En algunas realizaciones, los componentes del modelador de red de objetos se configuran para, tras completar una operación de lectura, disminuir el *rcnt* de la línea de índice en uno. Si *rcnt* llega a cero, el componente del modelador de redes de objetos que detecta esta condición se configura para transmitir el evento *rzero* a otros componentes del modelador de redes de objetos (por ejemplo, otro subproceso esperando para escribir), notificando así al componente en espera que puede continuar con su operación. El acceso atómico al *rcnt* de la línea de índice está garantizado a través del *vlock* de *lineraíl*.

En algunas realizaciones, los componentes del modelador de red de objetos (como un subproceso que ejecuta el gestor de índices) se configura para permitir que las operaciones de escritura en la matriz de ranuras de índice *arr* continúen siempre que ningún otro subproceso esté escribiendo en *arr* o esperando para escribir en *arr*. Si es necesario, los componentes del modelador de redes de objetos retrasan una operación de escritura hasta que se completa otra operación de escritura simultánea. Cuando un componente del modelador de red de objetos permite que continúe una operación de escritura, el *wreq* de la línea de índice se incrementa en uno y el *wbusycnt* de *lineraíl* también se incrementa en uno. En una realización, *wreq* y *wbusycnt* son ambos semáforos singleton y, por lo tanto, solo puede haber una operación de escritura simultánea por línea de índice y por *lineraíl*. A continuación, se adquiere el bloqueo *wbusy* y se verifica que la renta de la línea de índice sea cero. El acceso atómico a todas las variables de estado está garantizado a través del *vlock* de *lineraíl*. Si en este punto, la renta es distinta de cero, el componente del modelador de red de objetos de escritura (por ejemplo, hilo) se pone a dormir hasta que el último de los lectores actuales se haya completado. No se permiten nuevas operaciones de lectura porque el componente del modelador de red de objetos de escritura mantiene el bloqueo *wbusy*. El último subproceso en completar su lectura establece *rcnt* en cero y el componente de modelador de red de objetos de detección transmite el evento *rzero* al componente de modelador de red de objetos de escritura durmiente, que luego se activa y continúa con la operación de escritura.

Una vez completada una operación de escritura, los componentes del modelador de red de objetos de escritura reducen *wreq* de la línea de índice y *wbusycnt* de *lineraíl* y liberan el bloqueo *wbusy* de línea de índice. El acceso atómico se asegura con el *vlock* de *lineraíl*. En este punto, los componentes del modelador de redes de objetos han liberado su acceso exclusivo y todas las demás operaciones en otros componentes del modelador de redes de objetos pueden continuar.

En algunas realizaciones, una vez que un componente modelador de red de objetos ha adquirido la línea de índice para lectura, se garantiza que la matriz de ranuras de índice *arr* permanecerá consistente hasta que sea liberada por el componente modelador de red de objetos. En estas realizaciones, el componente del modelador de redes de objetos realiza la búsqueda escaneando *arr* linealmente comenzando en *rdx* y terminando justo antes de *wdx*. Los componentes del modelador de redes de objetos se configuran para ejecutar una operación de lectura como se describe a continuación, según el *modo de linerail*.

Si el *modo* del linerail = *PERMUTACIÓN*, el componente del modelador de red de objetos interpreta la ranura de índice como una estructura de *hamslot*, que se describe en la Tabla 5 a continuación. Durante la ejecución de la operación de lectura, el componente del modelador de redes de objetos visita todas las ranuras en la matriz de ranuras de *hamslot* secuencialmente, comenzando en *rdx* y terminando justo antes de *wdx*. Si el puntero del objeto de la ranura es NULO, lo que indica que la ranura se ha eliminado, el componente del modelador de red de objetos omite la ranura. De lo contrario, el componente del modelador de redes de objetos compara la huella digital de la ranura con la huella digital de sondeo. Si la distancia de hamming de la huella digital de sondeo a la huella digital de la ranura se encuentra dentro del umbral configurado, el componente del modelador de redes de objetos agrega el puntero del objeto de la ranura a una matriz de resultados de candidatos de recuperación.

TABLA 5

Tipo	Nombre	Descripción
FP_t	fp	Huella digital de 64 bits para una coincidencia aproximada (dentro de la distancia de hamming)
object_t*	object	Puntero a objeto en el almacén de objetos

Si el *modo* del linerail = *DIMENSIÓN*, el componente del modelador de red de objetos interpreta la ranura de índice como una estructura de *ranura* de dimensión, que se describe en la Tabla 6 a continuación. Durante la ejecución de la operación de lectura, el componente del modelador de redes de objetos visita todas las ranuras de la matriz de ranuras *dims* de forma secuencial, comenzando en *rdx* y terminando justo antes de *wdx*. Si el puntero del objeto de la ranura es NULO, lo que indica que la ranura se ha eliminado, el componente del modelador de red de objetos omite la ranura. De lo contrario, el componente del modelador de redes de objetos compara el identificador de dimensión de la ranura con el identificador de dimensión de sondeo. Si los dos se ajustan a una relación predefinida (por ejemplo, son iguales), el componente del modelador de redes de objetos agrega el puntero del objeto de la ranura a una matriz de resultados de candidatos de recuperación.

TABLA 6

Tipo	Nombre	Descripción
uint64_t	key	Dimensión de vector <i>n</i> para el índice de dimensión <i>n</i> -ésimo. (solo se usan 26 LSB).
object_t*	object	Puntero a objeto en el almacén de objetos

Si el *modo* del linerail = *OBID*, el componente del modelador de red de objetos interpreta el *indexslot* como una estructura *objslot*, que se describe en la Tabla 7 a continuación. Durante la ejecución de la operación de lectura, el componente del modelador de redes de objetos visita todas las ranuras de la matriz *objslot arr* secuencialmente, comenzando en *rdx* y terminando justo antes de *wdx*. Si el puntero del objeto de la ranura es NULO, lo que indica que la ranura se ha eliminado, el componente del modelador de red de objetos omite la ranura. De lo contrario, el componente del modelador de redes de objetos ejecuta uno de los siguientes procesos:

1. Si se especifica SUBID en el sondeo (solicitando así una sola manifestación de objeto), el componente del modelador de red de objetos compara el identificador de ID de objeto de *objslot obid_H* con los 64 bits superiores de la sonda OBID. Si el componente del modelador de red de objetos determina que los dos se ajustan a una relación predeterminada (por ejemplo, son iguales), entonces se elimina la referencia al objeto de puntero de *objeto* de *objslot* para comparar los 64 bits inferiores del OBID. Si el componente del modelador de redes de objetos determina que estos dos se ajustan a una relación predeterminada (por ejemplo, son iguales), entonces el puntero del objeto se vuelve a desreferenciar para comparar el SUBID del objeto con el SUBID de la sonda. Si el componente del modelador de redes de objetos determina que estos dos se ajustan a una relación predeterminada (por ejemplo, son iguales), se declara una coincidencia y se devuelve el resultado único en una matriz de objetos coincidentes terminada en centinela NULO.
2. Si no se especifica SUBID en el sondeo, el componente del modelador de red de objetos ejecuta el mismo proceso anterior, con la excepción de la coincidencia SUBID. El resultado es una matriz terminada en centinela NULO de cualquier objeto que coincida con el OBID del sondeo.

TABLA 7

Tipo	Nombre	Descripción
uint64_t	obid_H	64 bits superiores del OBID
object_t*	object	Puntero a objeto en el almacén de objetos

Si el *modo* del linerail = SUBID, el componente del modelador de red de objetos interpreta el indexslot como una estructura *objslot*, que se describe en la Tabla 8 a continuación. Durante la ejecución de la operación de lectura, el componente del modelador de redes de objetos visita todas las ranuras en la matriz de ranuras *obj arr* secuencialmente, comenzando en *rdx* y terminando justo antes de *wdx*. Si el puntero del objeto de la ranura es NULO, lo que indica que la ranura se ha eliminado, el componente del modelador de red de objetos omite la ranura. De lo contrario, el componente del modelador de red de objetos compara el identificador de ID de subobjeto de la ranura *obj obid_H* con los 64 bits superiores de la sonda SUBID. Si el componente del modelador de red de objetos determina que los dos se ajustan a una relación predeterminada (por ejemplo, son iguales), entonces se elimina la referencia al objeto del puntero del *objeto* de la ranura *obj* para comparar los 64 bits inferiores de los SUBID del objeto y la sonda. Si el componente del modelador de redes de objetos determina que estos dos se ajustan a una relación predeterminada (por ejemplo, son iguales), se declara una coincidencia y se devuelve el resultado único en una matriz de objetos terminada en centinela NULO que coinciden con el SUBID de sondeo.

TABLA 8

Tipo	Nombre	Descripción
uint64_t	obid_H	64 bits superiores del SUBID
object_t*	object	Puntero a objeto en el almacén de objetos

En algunas realizaciones, una vez que un componente del modelador de red de objetos (como un subproceso que ejecuta el gestor de índices) adquiere una línea de índice para escribir, se garantiza que otros subprocesos no leerán la matriz de ranuras de índice *arr*, lo que permite que se complete una operación de escritura sin el riesgo de que se produzcan búsquedas incoherentes. En estas realizaciones, el componente modelador de red de objetos ejecuta una operación de escritura de un objeto en el índice almacenando un puntero de objeto y un identificador de ID de objeto del objeto en la ranura de índice en *wdx*. El identificador de ID de objeto depende del *modo* tabla, como se describe anteriormente para las operaciones de búsqueda de línea de índice.

En algunas realizaciones, si *wdx* hace referencia a una ranura más allá del final de la matriz de ranuras de índice, el componente del modelador de red de objetos asigna una matriz más grande usando uno de los asignadores de matriz de ranuras de índice (consulte la Figura 29). A continuación, de acuerdo con estas realizaciones, el componente del modelador de redes de objetos copia la matriz antigua en la nueva matriz mientras también realiza la desfragmentación según sea necesario. Luego, el componente del modelador de redes de objetos desasigna la matriz anterior. En algunas realizaciones, el componente modelador de red de objetos también mantiene y actualiza los atributos de contabilidad en la línea de índice y el linerail según sea necesario.

En algunas realizaciones, una vez que un componente del modelador de red de objetos (como un subproceso que ejecuta el gestor de índices) adquiere una línea de índice para escribir, se garantiza que otros subprocesos no leerán la matriz de ranuras de índice *arr*, lo que permite que se complete una operación de eliminación sin el riesgo de que se produzcan búsquedas incoherentes. En estas realizaciones, el componente modelador de red de objetos ejecuta una operación de eliminación de un objeto del índice escaneando la matriz de ranuras de índice comenzando en *rdx* hasta que el puntero del objeto a eliminar coincida con uno en la matriz de ranuras de índice. Además, en estas realizaciones, si se encuentra el objeto a eliminar, el componente modelador de red de objetos establece el identificador de ID de objeto de la ranura de índice en cero y el puntero de objeto en NULO. Esto crea un "agujero" en la matriz de ranuras de índice que será ignorado por futuras operaciones de búsqueda. La operación de eliminación se completa una vez que se ha encontrado y eliminado un objeto coincidente. El escaneo secuencial se detiene cuando se alcanza *wdx*, lo que significa que el objeto al que se hace referencia no está indexado y no se realiza ninguna acción. En algunas realizaciones, el componente modelador de red de objetos también mantiene y actualiza los atributos de contabilidad en la línea de índice y el linerail según sea necesario.

En algunas realizaciones, si el número de objetos activos en la matriz de ranuras de índice cae por debajo de un umbral, el componente del modelador de red de objetos asigna una matriz más pequeña usando uno de los asignadores de matriz de ranuras de índice. A continuación, de acuerdo con estas realizaciones, el componente del modelador de redes de objetos copia la matriz antigua en la nueva matriz mientras también realiza la desfragmentación según sea necesario. Luego, el componente del modelador de redes de objetos desasigna la matriz anterior.

En algunas realizaciones, el modelador de redes de objetos se configura para operar usando varios índices completamente en la memoria. Para permitir un reinicio rápido del sistema después de un apagado, todos los índices pueden escribirse en archivos en el disco. El formato de archivo se muestra en la Figura 31. Como se ilustra, un archivo de índice tiene dos partes principales: un encabezado de índice y uno o más registros de índice.

En una realización ilustrada por la Figura 31, el encabezado del índice se divide en tres partes: parámetros de índice, máscaras y cambios de permutación de clave de índice, y descriptor de asignador de matriz de ranuras de índice. Los parámetros de índice reflejan muchos de los parámetros de los rieles lineales, omiten atributos de tiempo de ejecución como bloqueos, semáforos y punteros, y agregan marcadores de coherencia como la versión de API (*api_ver*) y el contador de operaciones (*opcnt*). Las máscaras de permutación de clave de índice y los turnos 404 reflejan los del atributo *perm* del *linrail*. El descriptor del asignador de matriz contiene información sobre los requisitos de memoria en tiempo de ejecución para este *linrail*. Esta información permite que el modelador de redes de objetos asigne previamente la cantidad exacta de memoria necesaria en el inicio para optimizar el rendimiento y minimizar la fragmentación.

En una realización ilustrada por la Figura 31, los registros de índice son un reflejo directo de la matriz de ranuras de índice en cada línea de índice para el *linrail*. En esta realización, solo se almacenan los objetos activos, con un contador de objetos para cada uno que indica el número de objetos activos por línea de índice. En algunas realizaciones, un objeto se representa mediante dos números en la estructura *idxrecbuf*: el número de bloque del almacén de objetos donde existe el objeto y el desplazamiento de bloque a la ranura en el bloque que contiene el objeto. En estas realizaciones, la referencia del objeto es válida solo si el *opcnt* del encabezado del índice coincide con el *opcnt* del almacén de *objetos*. Si al iniciar estos no coinciden, el modelador de redes de objetos rechaza el archivo de índice y vuelve a generar el archivo de índice desde el almacén de objetos, lo que resulta en un inicio más lento.

Almacén de objetos

La Figura 32 ilustra componentes de datos que se incluyen dentro de un almacén de objetos, como el almacén de objetos 214 descrito anteriormente con referencia a la Figura 2. En algunas realizaciones, el almacén de objetos se organiza como una estructura raíz *objectstore_t*, que se describe a continuación en la Tabla 9, que hace referencia a un conjunto de bloques de objetos *objblock_t*, que se describe a continuación en la Tabla 10. En estas realizaciones, el almacén de objetos tiene una capacidad de objetos máxima fija y una relación uno a uno con un archivo de partición en el disco. Además, en estas realizaciones, el almacén de objetos puede incluir hasta 65 536 bloques de objetos, cada uno con una capacidad fija de 262 144 objetos para una capacidad teórica total de aproximadamente 17 billones de objetos. Sin embargo, se aprecia que otras realizaciones pueden tener diferentes capacidades y las realizaciones no están limitadas a una capacidad fija o variable particular.

TABLA 9

Tipo	Nombre	Descripción
long long	s z	Capacidad total actual de objetos, es decir, número de ranuras de objetos asignados en todos los bloques de objetos
long long	nobj	Número total de objetos activos
uint64_6	opcnt	Número total de operaciones de modificación ejecutadas
Int	vector_max_size	Número máximo de elementos vectoriales admitidos en V_{int}
char []	path	Directorio raíz en el disco para almacenar datos de objetos
char []	masterpath	Nombre de archivo para el registro maestro de Simarray
char []	collecpath	Nombre de archivo para el registro del conjunto de colecciones de Simarray
Int	open_part_s	Número de archivos de partición de bloque abiertos
objblock_t*	head	Puntero al primer bloque de objetos
objblock_t*	wblock	Puntero al bloque de objeto utilizado para escribir nuevos objetos
objblock_t*	tail	Puntero al último bloque de objeto preasignado
object_t**	block_objects	Matriz de punteros a objetos indexados por desplazamiento de objeto
blockmaster_t []	master	Matriz de registros maestros con tamaño de retención e índices de escritura por bloque
collection set t	collections	Conjunto de filtros de acceso a la colección de Objectstore
FILE*	change log	Archivo de registro de confirmación actual en uso para escribir actualizaciones/eliminaciones
size_t	commit_cycle_count	Número de operaciones desde la última confirmación
size_t	last_pending_count	Número de operaciones pendientes restantes después de la confirmación
Long	block_max_dirty	Número máximo de objetos modificados en bloque antes de vaciar
long long	nops_pending	Número de operaciones actualmente pendientes en el registro de confirmación

En algunas realizaciones, un gestor de objetos, como el gestor de objetos 210 descrito anteriormente con referencia a la Figura 2, se configura para crecer y reducir dinámicamente el almacén de objetos en incrementos de un *objblock*

según corresponda. En estas realizaciones, un objblock tiene un conjunto fijo y preasignado de ranuras de objetos que se activan a medida que los datos del objeto se incorporan al modelador de red de objetos 202. Cuando el bloque actual en el que está escribiéndose está lleno, el siguiente bloque puede convertirse en el bloque de escritura y puede crearse si aún no se ha asignado previamente.

En algunas realizaciones, el gestor de objetos implementa un sistema de permisos de objetos para permitir que un modelador de redes de objetos, como el modelador de redes de objetos 202 descrito anteriormente con referencia a la Figura 2, para mantener conjuntos de datos segmentados. El proceso de indexación y búsqueda que se describe a continuación dentro de la sección Índices filtra automáticamente los objetos por un atributo de segmento según el contexto actual de la colección de indexación/búsqueda.

Tabla 10

Tipo	Nombre	Descripción
uint16_t	Id	ID de bloque 0,1,2,...,65 535
long	s z	Capacidad del bloque en número de objetos almacenables
long	nobj	Número de <i>objetos</i> activos actuales en el bloque
long	Wdx	Escribir índice, desplazamiento en objetos para la siguiente ranura disponible
long	ndirty	Número de objetos modificados no escritos en el disco
int	defer	Si se establece, las escrituras en disco se diferencian, se requiere compromiso explícito
int	vector max size	Elementos máximos permitidos en V_{int}
vecelem_t*	nullvecAn	vector vacío, es decir, 0
long long	vecrec offset zero	Compensación de bytes al primer registro vectorial en el archivo de datos en el disco
long long	metarec_offset_zero	Compensación de bytes al primer metaregistro en el archivo de datos en el disco
uint8_t	set no	Número de conjunto de directorio para el archivo de datos en el disco
char []	vecfname	Ruta completa al archivo de datos de bloque en el disco
FILE*	vecpart	Puntero de archivo de datos de bloque
char []	metaf_name	Ruta completa al archivo de metadatos en el disco
FILE *	metapart	Puntero de archivo de metadatos
Long	wdx_last	Último wdx escrito para
time_t	t_last	Marca de tiempo en la que se escribió el archivo de datos en el disco
objblock_t*	prey	Puntero al bloque anterior
objblock_t*	next	Puntero al siguiente bloque
object_t []	objects	Matriz de ranuras de objetos preasignados en bloque

En algunas realizaciones, la matriz de *objetos* en *objblock* es un gran conjunto de objetos preasignados de tipo *object_t*, que se describe a continuación en la Tabla 11. Estos objetos son los objetos Simarray discutidos a lo largo de la presente descripción.

Tabla 11

Tipo	Nombre	Descripción
operation_t	Operation	Id. de operación de 64 bits que modificó por última vez este objeto
FP_t	Fp	Huella digital V_{fp}
vecelem_t*	Vector	Vector interno V_{int}
objectid_t	Obid	ID del objeto OBID
objectid_t	Subid	ID de referencia SUBID
locator_t	Locator	Varios metadatos de objetos
accessor_t	Accessor	Filtro de permisos de acceso a objetos
Long	Cardinality	Número de referencias entrantes a este objeto
long long	Metaoffset	Compensación de bytes en el archivo de datos en el disco, que contiene metadatos para el objeto
flextypet_t	Flex	Metadatos in-object flexibles y multipropósito (pequeños)
objblock_t*	Pblock	Puntero de vuelta al bloque que posee este objeto

En algunas realizaciones, los objetos que se insertan en el almacén de objetos se escriben en el *objblock* basado en memoria y en un archivo de datos de bloque en el disco. El formato de archivo de datos de bloque se muestra en la Figura 33. Como se ilustra, el formato de archivo de datos en bloque incluye un encabezado seguido de uno o más registros que almacenan datos de objetos. En estas realizaciones, el gestor de objetos escribe nuevos objetos en el disco a medida que se crean en la memoria. Además, en estas realizaciones, el gestor de objetos protege los objetos

modificados en el disco inmediatamente (el valor predeterminado) o difiere la protección de las actualizaciones por lotes en el disco en respuesta a la recepción de un comando de compromiso explícito. Al proteger los objetos modificados en el disco, el gestor de objetos transmite los objetos modificados a un registro de cambios que rota continuamente. De esta forma, el gestor de objetos garantiza escrituras secuenciales en el disco la mayor parte del tiempo (para mejorar el rendimiento) con actualizaciones por lotes ocasionales de las partes de datos del bloque de objetos en el disco.

En algunas realizaciones, los objetos pueden contener opcionalmente metadatos "BLOB" arbitrarios, es decir, cualquier dato binario de hasta 2 GB. En estas realizaciones, los metadatos no se almacenan en la memoria. Más bien, en estas realizaciones, los metadatos se almacenan en un archivo de metadatos asociado con el archivo de bloque de objetos. Por lo tanto, en estas realizaciones, una partición de datos en bloque tiene dos archivos: el archivo de datos de objetos y el archivo de metadatos.

En algunas realizaciones, un componente del modelador de redes de objetos, como un motor de consultas (por ejemplo, el motor de consultas 218 descrito anteriormente con referencia a la Figura 2) o el gestor de objetos, almacena y accede a los metadatos correspondientes a un objeto usando el atributo *metaoffset* del objeto. En estas realizaciones, cuando *el metaoffset* es cero, no existen metadatos para el objeto. Además, en estas realizaciones, cuando los metadatos no son cero, el atributo *metaoffset* interpretado por el componente modelador de red de objetos como un desplazamiento en el archivo de metadatos donde se encuentra el registro de metadatos para el objeto. Además, de acuerdo con estas realizaciones, un registro de metadatos incluye un conjunto fijo de metaatributos estándar que identifican el objeto (si es necesario para la recuperación), seguido de una matriz de bytes variable que contiene los metadatos reales. La Figura 34 ilustra el formato de un archivo de metadatos de acuerdo con una realización.

En algunas realizaciones, cuando se accede a los metadatos mediante una operación de búsqueda, como por el motor de consultas, primero se abrirá el archivo de metadatos (si aún no está abierto) y el puntero del archivo se colocará en bytes de *metaoffset* desde el principio. Luego, los datos se leen del archivo y se devuelven. La sobrecarga de rendimiento del uso de metadatos depende de la velocidad del subsistema de E/S del disco.

Asignadores

Debido al amplio uso de la memoria, algunas realizaciones del modelador de redes de objetos implementan asignadores de memoria personalizados para los datos vectoriales V_{int} que se usan en los objetos y las matrices de ranuras de índice que se usan en los rieles lineales. Los diagramas de alto nivel para los asignadores de vectores (como uno o más asignadores de vectores descritos anteriormente con referencia a la Figura 29) y los asignadores de índices (como uno o más asignadores de índices descritos anteriormente con referencia a la Figura 29) se muestran en las Figuras 35 y 36, respectivamente. En algunas realizaciones, los asignadores están configurados para preasignar bloques de memoria adecuados para vectores y para conjuntos de ranuras de índice. En estas realizaciones, solo se admite un conjunto seleccionado de tamaños para cada tipo de asignador. Sin embargo, las realizaciones no se limitan a los tamaños particulares mencionados expresamente en la presente descripción.

En algunas realizaciones, cuando se representan vectores internos V_{int} , el gestor de objetos cuantificará el tamaño del vector para que quepa en la matriz de vectores más pequeña posible usando el asignador mínimo A_k como se describe en la sección Codificación de vectores en la presente descripción. En estas realizaciones, hay 20 asignadores de vectores que admiten vectores de los siguientes tamaños: 4, 8, 12, 16, 20, 32, 48, 64, 80, 96, 128, 160, 192, 224, 256, 320, 384, 448, 512, y 520.

Por ejemplo, para representar un vector con 27 elementos, el gestor de objetos usaría el sexto asignador para asignar V_{int} con capacidad 32, dejando 5 elementos sin usar al final. La cantidad de espacio desperdiciado promedio se ha equilibrado con los beneficios de tener un pequeño conjunto de asignadores de tamaño fijo para optimizar el rendimiento y minimizar la fragmentación con la sobrecarga de administración de memoria asociada.

En algunas realizaciones, el gestor de índices se configura para asignar matrices de ranuras de índice utilizadas para indexar objetos como parte de las estructuras de rieles lineales a partir de un conjunto de asignadores de ranuras de índice. Las matrices de ranuras de índice crecen y se reducen dinámicamente con el funcionamiento normal del modelador de redes de objetos. Si el gestor de índices usara un asignador genérico (por ejemplo, *malloc/free*), los efectos de la fragmentación de la memoria serían perjudiciales para un funcionamiento eficiente. En su lugar, el gestor de índices usa asignadores personalizados específicos de la aplicación. De esta forma, el gestor de índices crea y mantiene grandes bloques de memoria que se ajustan con precisión para un propósito específico.

En estas realizaciones, hay 48 asignadores de ranuras de índice distintos que admiten matrices de ranuras de índice de los siguientes tamaños: 2, 12, 48, 104, 171, 245, 324, 408, 494, 584, 676, 770, 865, 963, 1062, 1163, 1265, 1368, 1472, 1577, 1683, 1790, 1898, 2007, 2117, 228, 2339, 2451, 2563, 2676, 2790, 2605, 3020, 3135, 3252, 3368, 3485, 3603, 3721, 3840, 3959, 4078, 4198, 4319, 4440, 4561, 4682, 4804. Cuando una matriz de ranuras de índice se queda sin espacio, el gestor de índices usa el siguiente asignador superior para reemplazar la matriz anterior con una matriz nueva. Por ejemplo, si la matriz actual se asignó usando el asignador "12" y debe escribirse el elemento trigésimo, entonces el gestor de índices usará el asignador "48" para proporcionar una nueva matriz de ranuras de índice para

reemplazar a la anterior a la que se hace referencia desde la estructura de línea de índice. El gestor de índices copia los datos de la matriz anterior en la matriz nueva y desfragmenta los datos si es necesario. Luego, el gestor de índices desasigna la matriz anterior y la pone a disposición del asignador "12" para uso futuro. Tenga en cuenta que no se usará malloc ni free en este proceso. Del mismo modo, si el gestor de índices elimina objetos de un índice, la matriz anterior se reemplazará por una más pequeña siempre que sea posible. En algunas realizaciones, los propios asignadores se expandirán y contraerán según sea necesario en bloques grandes a intervalos raros. Esto puede requerir llamadas poco frecuentes a malloc y free, pero en porciones muy grandes. De este modo, se minimiza la fragmentación de la memoria.

Los asignadores que aún no se han utilizado para proporcionar una matriz o un vector de ranura de índice no ocupan espacio, aparte de una cantidad muy pequeña de metaatributos usados para la contabilidad. Por ejemplo, si ninguna matriz de ranuras de índice en ningún linerail tiene más de 48 ranuras, todos los asignadores por encima de 48 estarán vacíos y no ocuparán ningún espacio preasignado.

Se aprecia que la matriz de ranuras de índice puede crecer más que el asignador más grande ("4804"). Esto puede suceder en situaciones en las que los valores hash de los datos de entrada no se distribuyen uniformemente. La indexación de dimensiones es propensa a este comportamiento si algunas dimensiones vectoriales ocurren con una frecuencia desproporcionada en comparación con otras. Un vectorizador, como el vectorizador 204 descrito anteriormente con referencia a la Figura 2, con una normalización agresiva de IDF puede ser deseable en tales casos. Sin embargo, algunas realizaciones del modelador de redes de objetos permiten matrices de ranuras de índice de cualquier tamaño. En estas realizaciones, el gestor de índices utiliza malloc y free para asignar memoria para matrices de ranuras de índice que son demasiado grandes para un asignador de índices estándar. Sin embargo, también se aprecia que el rendimiento de búsqueda puede verse afectado cuando se usan matrices de gran tamaño. Por lo tanto, en algunas realizaciones, el gestor de índices registra y proporciona estadísticas del perfil del asignador de ranuras de índice para guiar la optimización en estas situaciones.

Representaciones multivectoriales

En algunas realizaciones, el modelador de redes de objetos se configura para representar objetos externos grandes (por ejemplo, documentos) como un conjunto de características que es una forma eficiente de normalizar la información a través de la reducción de la dimensionalidad. Sin embargo, en estas realizaciones, puede que no sea posible crear un único conjunto de características para describir correctamente el objeto original.

Por ejemplo, en un ejemplo, el modelador de redes de objetos puede usarse para modelar documentos de texto como objetos. En este ejemplo, representar un libro completo como un vector de características puede ser difícil para el vectorizador. Para abordar el problema, el modelador de redes de objetos puede representar objetos desde varios "puntos de vista" usando múltiples vectores. De esta forma, el modelador de redes de objetos puede asociar objetos basándose en coincidencias parciales entre aspectos seleccionados de objetos.

La Figura 37 muestra cuatro vectores que representan un documento. Los vectores pueden combinarse en un centroide para hacer coincidir otros objetos o usarse individualmente. El modelador de redes de objetos permite que la aplicación use cualquier combinación de criterios de coincidencia de vectores para formar redes. Los objetos pueden representarse como múltiples vectores donde cada vector se basa en diferentes tipos de información. Algunos otros ejemplos incluyen:

1) **Segmentación lineal.** Pueden crearse vectores separados en diferentes partes de los datos de entrada. El nivel de granularidad es arbitrario. Podría ser un vector por capítulo en un libro, un vector por párrafo o algo más. También podrían ser segmentos de sonido, voz o video representados apropiadamente por un vectorizador de medios adecuado. Puede aplicarse una función de ventana a la entrada que permite la superposición de regiones de información por vector.

2) **Tipo de entidad** Pueden crearse vectores separados para diferentes categorías de información. Por ejemplo, podrían usarse vectores separados para representar personas, lugares, productos, empresas, medicamentos, alimentos, etc. Los vectorizadores especializados que reconocen entidades de tipos específicos pueden ser útiles en este escenario.

3) **Clase de palabras.** Pueden usarse vectores separados para representar sustantivos, verbos y adjetivos en el documento de entrada. Además, los vectores pueden crearse alrededor de segmentos de texto basados en el tiempo verbal (pasado, presente, futuro), o alrededor de adjetivos para vectores basados en sentimientos.

4) **Cubo de palabras.** Para mejorar la solidez frente a los falsos negativos, pueden usarse varios vectores en los que cada vector solo considera un subconjunto de palabras, por ejemplo, basándose en grupos de primeras letras como A-F, G-L, M-R, S-Z. Esto es útil para aplicaciones de detección de duplicados de alta recuperación basadas solo en el índice de permutación (para el rendimiento) donde la fragilidad de las huellas digitales puede ser un problema.

Implementación de la unidad de procesamiento de gráficos (GPU)

Las GPU son adecuadas para realizar una operación en muchos datos simultáneamente. Esto se logra empaquetando muchos núcleos "reducidos" en la GPU, cada uno de los cuales puede realizar operaciones independientes simultáneamente. Además, cada uno de estos núcleos contiene múltiples ALU (unidades lógicas aritméticas), lo que permite que un núcleo aplique una operación a múltiples datos en un solo ciclo de reloj. Esta funcionalidad se denomina procesamiento SIMD (datos múltiples de instrucción única).

Con referencia a la arquitectura de muestra que se ilustra en la Figura 38, el bloque de obtención/decodificación gestiona la dispersión de una operación idéntica a todas las ALU. Cada ALU hace referencia a su bloque de contexto local para obtener los datos en los que debe realizarse la operación. Luego, los resultados se escriben en un búfer de salida, que puede no estar necesariamente ubicado en el bloque de contexto local (podría estar ubicado en el bloque de datos de contexto compartido).

Una GPU contiene varios núcleos y el proceso descrito anteriormente puede realizarse en cada uno de ellos. Por lo tanto, la cantidad de operaciones simultáneas que puede realizar una GPU generalmente se define multiplicando el recuento de núcleos y ALU por recuento de núcleos.

En algunas realizaciones, el modelador de redes de objetos puede implementarse usando una o más GPU de propósito general. En particular, las siguientes operaciones pueden ser codificación vectorial mejorada, generación de huellas digitales, comparación de huellas digitales y cálculo de similitud de coseno. Además, las realizaciones que se ejecutan en una GPU pueden implementar una huella digital más amplia, lo que permite una mayor resolución por dimensión.

Con respecto a la codificación vectorial, las operaciones de hash de características y normalización de peso pueden descargarse a la GPU en lotes, aprovechando el hecho de que cada hash es una operación independiente. Por lo tanto, pueden calcularse miles de hashes simultáneamente usando la GPU, lo que debería resultar en una mejora considerable de la velocidad al indexar grandes conjuntos de datos. Con respecto a la generación de huellas digitales, miles de operaciones de generación de fragmentos de huellas digitales pueden realizarse simultáneamente usando una GPU. En combinación con la codificación vectorial basada en GPU, el rendimiento de la indexación de datos debería mejorar considerablemente. Con respecto a la comparación de huellas digitales, en lugar de realizar escaneos de tablas lineales, los cálculos de distancia de hamming pueden realizarse en lotes en una GPU. Con respecto al cálculo de similitud de coseno, los productos de punto pueden calcularse en lotes en una GPU.

Con respecto a las huellas digitales anchas, en algunas realizaciones, las huellas digitales tienen un ancho de 64 bits para que las CPU x86-64 puedan procesarlas de manera eficiente. Al descargar la generación de huellas digitales y la comparación con una GPU, el sistema puede procesar huellas digitales más amplias. Dado que las GPU son muy buenas para procesar datos en paralelo, una huella digital más ancha (por ejemplo, 128 bits) puede dividirse en partes más pequeñas (32/64 bits, según la arquitectura de la GPU) que pueden procesarse como parte de un trabajo por lotes. De esta forma, puede procesarse una huella digital más ancha en el mismo tiempo que se tardaría en procesar una huella digital de 64 bits. Una huella digital más ancha permite más fragmentos, lo que a su vez se traduce en una mayor robustez frente a los cambios en el vector de entrada V_{int} .

Habiendo descrito así varios aspectos de al menos un ejemplo, se apreciará que a los expertos en la técnica se les ocurrirán fácilmente diversas alteraciones, modificaciones y mejoras. Por ejemplo, los ejemplos descritos en la presente descripción también pueden usarse en otros contextos. Dichas alteraciones, modificaciones y mejoras pretenden incorporar la presente invención cuando permanecen dentro del alcance de las siguientes reivindicaciones. En consecuencia, la descripción y los dibujos anteriores son sólo a modo de ejemplo.

REIVINDICACIONES

1. Un sistema informático (102) para identificar objetos dentro de una red de objetos que son suficientemente similares a una consulta recibida de una aplicación (208), el sistema informático (102) que comprende:

memoria (112) que comprende instrucciones ejecutables; y
un procesador (110) conectado operativamente a la memoria, teniendo el procesador una arquitectura que utiliza una palabra, el procesador se configura para ejecutar las instrucciones ejecutables con el fin de efectuar un método que comprende:

obtener un objeto de sondeo (Q) representativo de la consulta;
generar un vector externo (Q_{ext}) representativo del objeto de sondeo (Q) mediante un vectorizador (204);
convertir el vector externo (Q_{ext}) representativo del objeto de sondeo (Q) en un vector interno (Q_{int}) representativo del objeto de sondeo (Q) mediante el hash de características, normalización de peso y cuantificación de longitud;

generar una huella digital (Q_{fp}) representativa del objeto de sondeo basada en el vector interno (Q_{int}) representativo del objeto de sondeo (Q), en donde cada vector interno (Q_{int}) representativo del objeto de sondeo (Q) comprende una pluralidad de dimensiones, y en donde cada dimensión de la pluralidad de dimensiones está asociada con una magnitud respectiva, y en donde la generación incluye:

para el vector interno (Q_{int}) representativo del objeto de sondeo, generar una pluralidad de fragmentos de huellas digitales (P_j) representativos del vector interno (Q_{int}), en donde cada fragmento de huellas digitales (P_j) de la pluralidad de fragmentos de huellas digitales (P_j) se genera en base a una combinación de resúmenes de una pluralidad de dimensiones del vector interno (Q_{int}) y magnitudes asociadas respectivas con cada una de la pluralidad de dimensiones del vector interno (Q_{int}); y

agregar la pluralidad de fragmentos de huellas digitales generados (P_j) para generar la huella digital (Q_{fp}) representativa del objeto de sondeo, la huella digital generada (Q_{fp}) que tiene una dimensionalidad reducida en relación con el vector interno (Q_{int}) representativo del objeto de sondeo, en donde la huella digital se genera para encajar en un múltiplo par de palabras en la arquitectura del procesador;

realizar la siguiente comparación para identificar un primer conjunto de objetos candidatos:

comparar la huella digital (Q_{fp}) representativa del objeto de sondeo (Q) con una pluralidad de huellas digitales (V_{fp}) representativas de una pluralidad de objetos dentro de la red de objetos, en donde comparar la huella digital (Q_{fp}) representativa del objeto de sondeo con el pluralidad de huellas digitales (V_{fp}) representativas de la pluralidad de objetos dentro de la red de objetos comprende: calcular distancias de hamming respectivas entre la huella digital (Q_{fp}) representativa del objeto de sondeo y cada una de las huellas digitales (V_{fp}) representativas de la pluralidad de objetos dentro de la red de objetos; e identificar, como el primer conjunto de objetos candidatos, aquellos objetos de la pluralidad de objetos cuyas huellas digitales (V_{fp}) están dentro de una distancia de hamming predefinida desde la huella digital (Q_{fp}) representativa del objeto de sondeo;

calcular las primeras métricas de similitud respectivas entre el objeto de sondeo (Q) y cada uno de los objetos en el primer conjunto de objetos candidatos comparando el vector interno (Q_{int}) representativo del objeto de sondeo (Q) con un vector interno correspondiente (V_{int}) representativo de cada objeto en el primer conjunto de objetos candidatos, en donde la primera métrica de similitud comprende la similitud de coseno entre, o el índice de Jaccard, o una combinación de la similitud de coseno entre y el índice de Jaccard del vector interno (Q_{int}) representativo del objeto de sondeo (Q) y el correspondiente vector interno (V_{int}) representativo del objeto en el primer conjunto de objetos candidatos; y

generar un segundo conjunto de objetos candidatos basado en el primer conjunto de objetos candidatos, en donde el segundo conjunto de objetos candidatos incluye solo aquellos objetos cuyas primeras métricas de similitud superan un primer umbral predefinido y, por lo tanto, cuyos vectores internos (V_{int}) se definen como similares al vector interno (Q_{int}) representativo del objeto de sondeo;

calcular las segundas métricas de similitud respectivas entre el objeto de sondeo (Q) y cada uno de los objetos en el segundo conjunto de objetos candidatos comparando el vector externo (Q_{ext}) representativo del objeto de sondeo (Q) con un vector externo correspondiente (V_{ext}) representativo de cada objeto en el segundo conjunto de objetos candidatos; y generar un tercer conjunto de objetos candidatos basado en el segundo conjunto de objetos candidatos, en donde el tercer conjunto de objetos candidatos incluye solo aquellos objetos cuyas segundas métricas de similitud exceden un segundo umbral predefinido, y en donde la segunda métrica de similitud comprende la similitud de coseno entre, o el índice de Jaccard, o una combinación de la similitud de coseno entre y el índice de Jaccard, del vector externo representativo del objeto de sondeo (Q) y el vector externo correspondiente representativo del objeto en el segundo conjunto de objetos candidatos; y

devolver el tercer conjunto de objetos candidatos a la aplicación (208).

2. El sistema informático (102) de la reivindicación 1, en donde el primer umbral predefinido y el segundo umbral predefinido son iguales.

3. El sistema informático (102) de la reivindicación 1, en donde identificar el primer conjunto de objetos candidatos comprende además comparar el vector interno (Q_{int}) representativo del objeto de sondeo (Q) con una pluralidad

de vectores internos (V_{int}) representativos de la pluralidad de objetos dentro de la red de objetos; en donde cada vector interno (V_{int}) comprende una pluralidad de dimensiones, y en donde comparar el vector interno (Q_{int}) representativo del objeto de sondeo (Q) con la pluralidad de vectores internos (V_{int}) representativos de la pluralidad de objetos dentro de la red de objetos comprende:

para el vector interno (Q_{int}) representativo del objeto de sondeo (Q), comparar un número predefinido de dimensiones del vector interno (Q_{int}) representativo del objeto de sondeo (Q) con las dimensiones correspondientes de la pluralidad de vectores internos (V_{int}) representativo de la pluralidad de objetos dentro de la red de objetos.

4. El sistema informático (102) de la reivindicación 1, en donde el procesador (110) se configura para ejecutar las instrucciones ejecutables con el fin de efectuar el método que comprende además:
eliminar, del primer conjunto de objetos candidatos, cualquier instancia duplicada de objetos dentro del primer conjunto de objetos candidatos de modo que solo haya una única instancia de cada objeto dentro del primer conjunto de objetos candidatos.

5. El sistema informático (102) de la reivindicación 1, en donde el vector externo (Q_{ext}) representativo del objeto de sondeo comprende una pluralidad de características que describen el objeto de sondeo (Q), y en donde cada característica de la pluralidad de características está asociada con un respectivo peso.

6. El sistema informático (102) de la reivindicación 5, en donde generar el vector interno (Q_{int}) representativo del objeto de sondeo (Q) basado en el vector externo (Q_{ext}) representativo del objeto de sondeo (Q) comprende:

aplicar una función hash a cada característica de la pluralidad de características del vector externo (Q_{ext}) representativo del objeto de sondeo (Q);

aplicar al menos una función de escala y una función de normalización a cada peso asociado con cada característica de la pluralidad de características del vector externo (Q_{ext}) representativo del objeto de sondeo (Q); y

en donde el vector interno (Q_{int}) representativo del objeto de sondeo (Q) se representa en un número fijo de bits, y en donde el vector externo correspondiente (Q_{ext}) representativo del objeto de sondeo (Q) se representa en un número variable de bits.

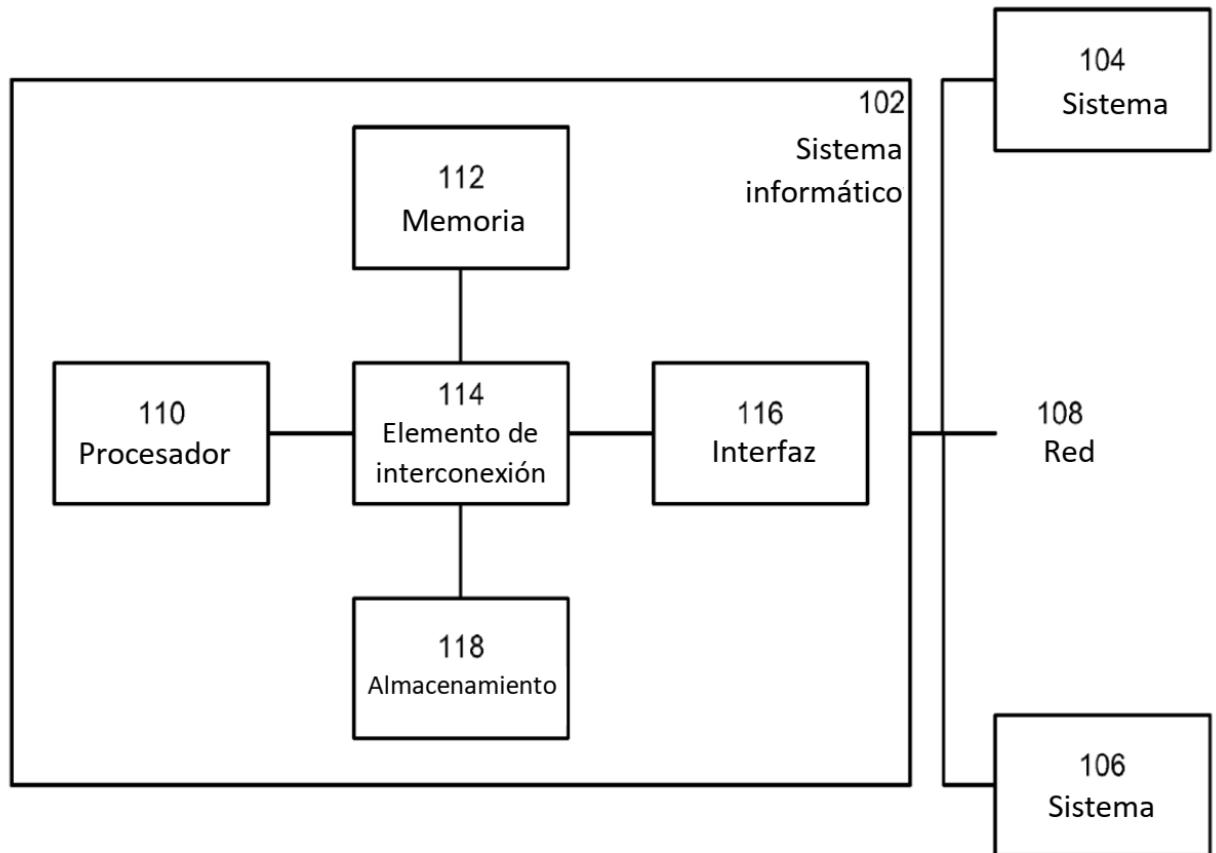


FIGURA 1

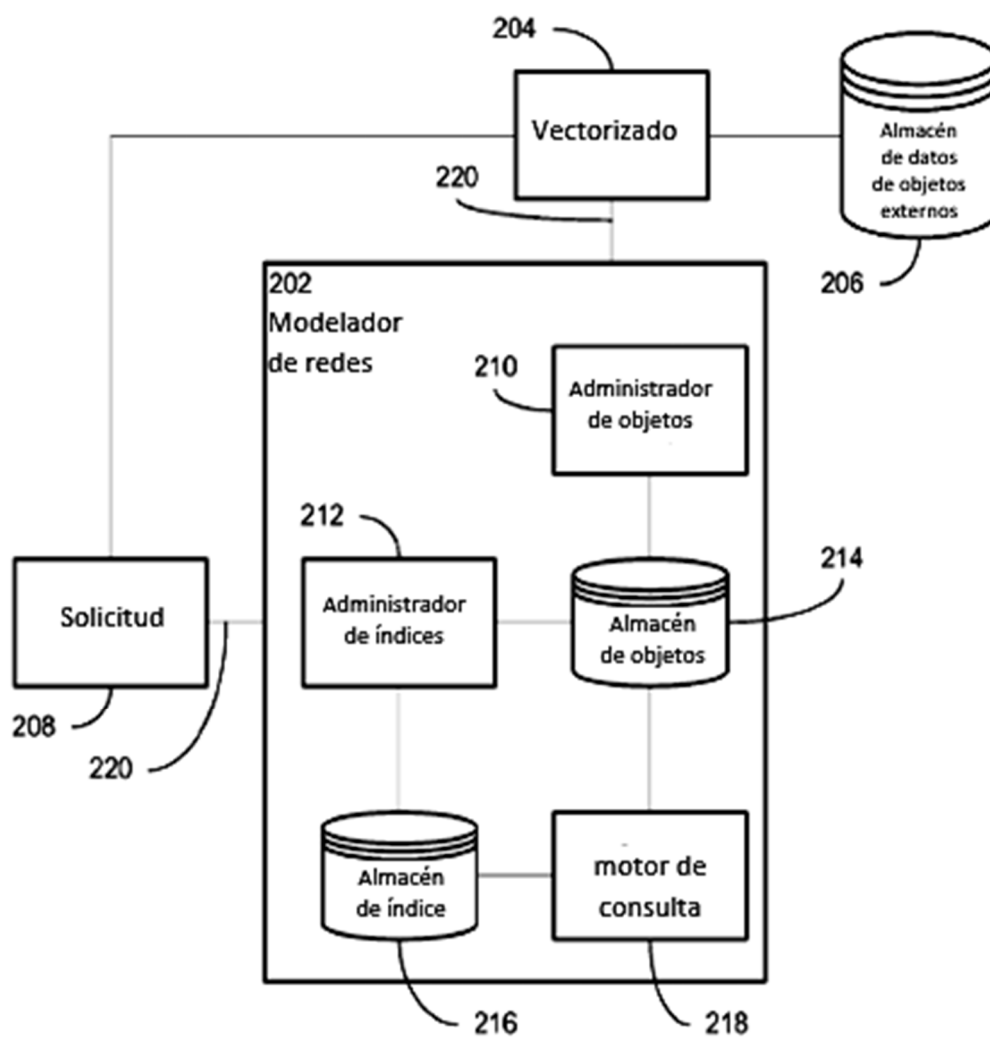


Figura 2

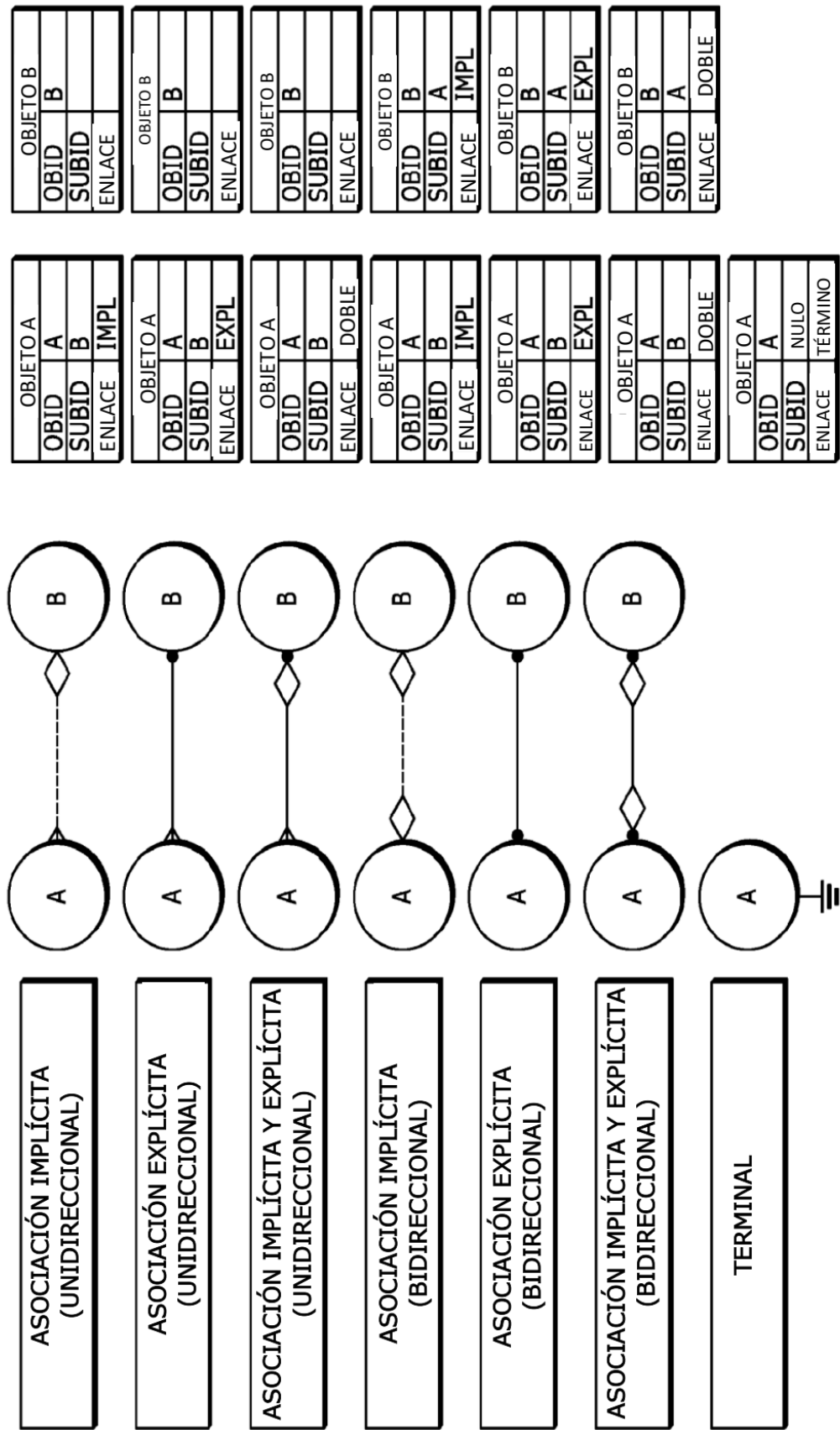


FIGURA 3

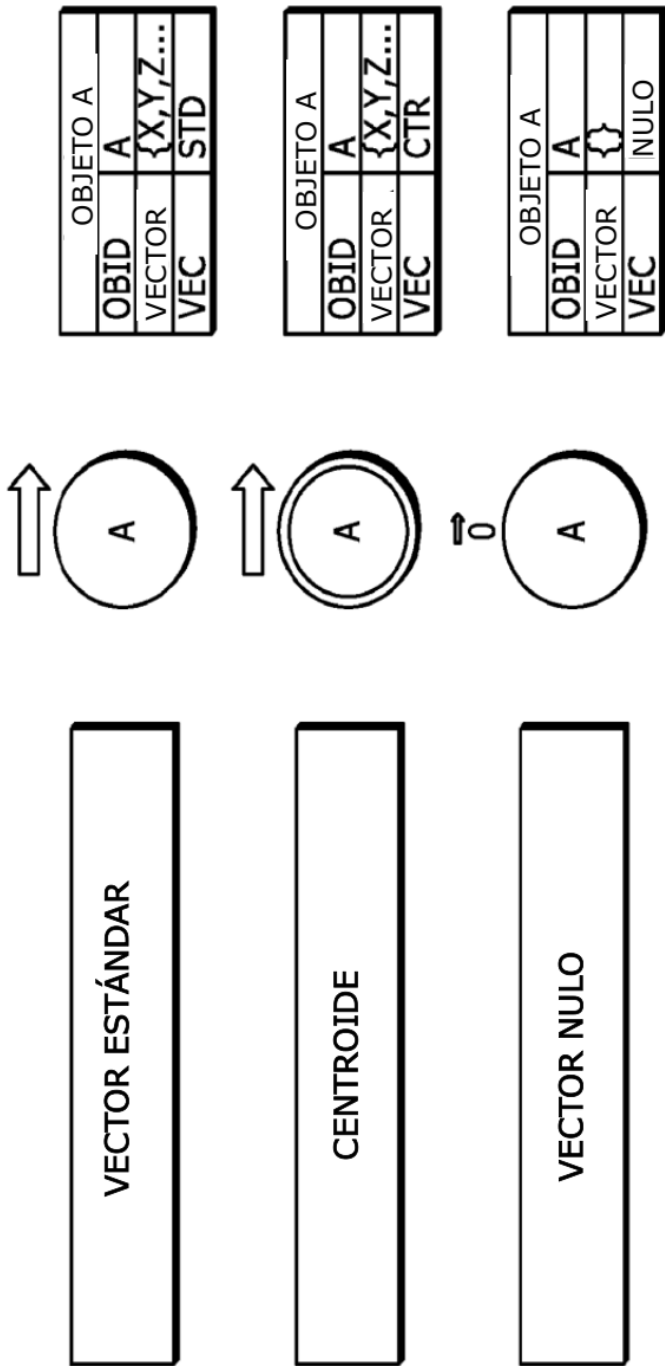


FIGURA 4

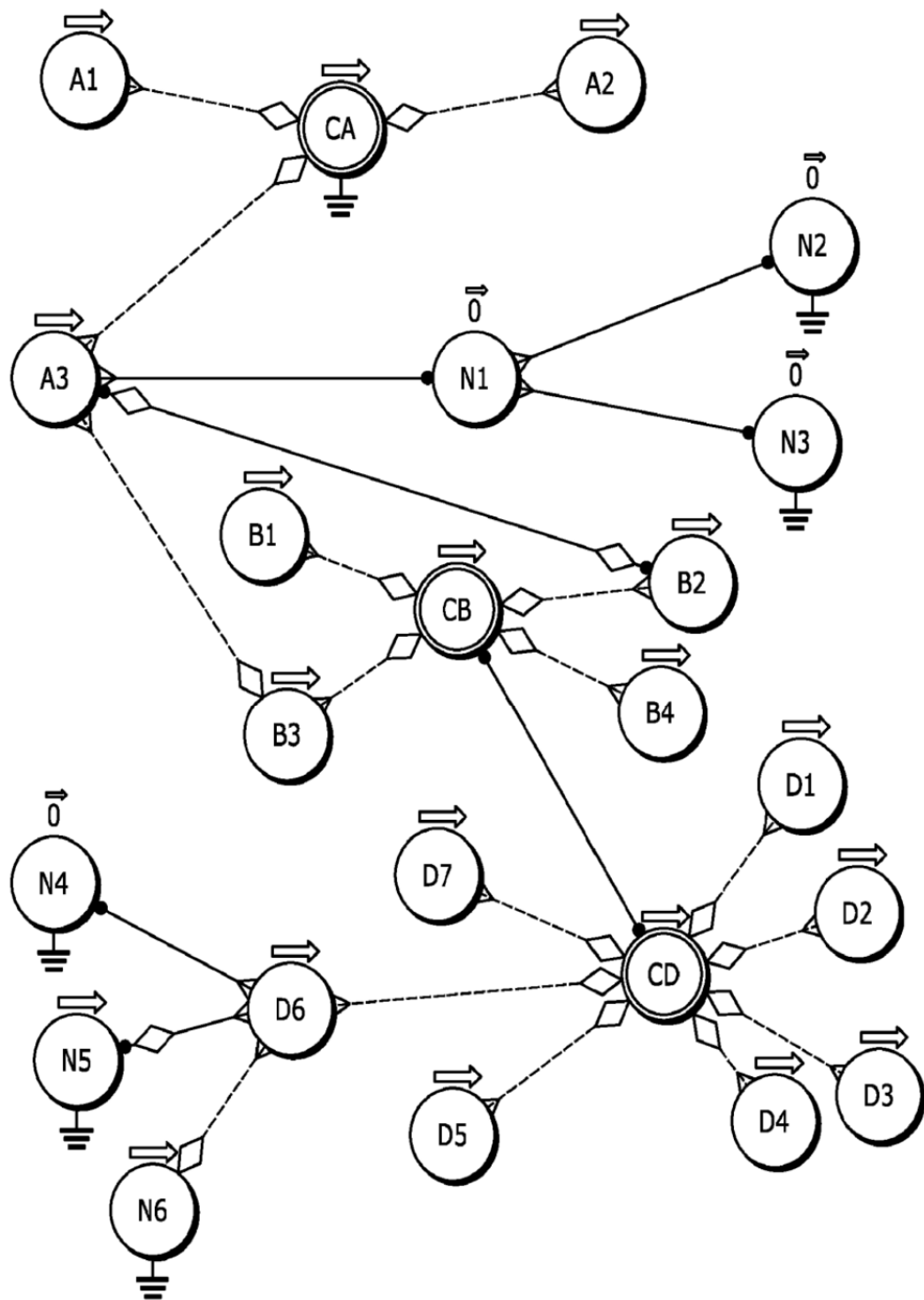


FIGURA 5

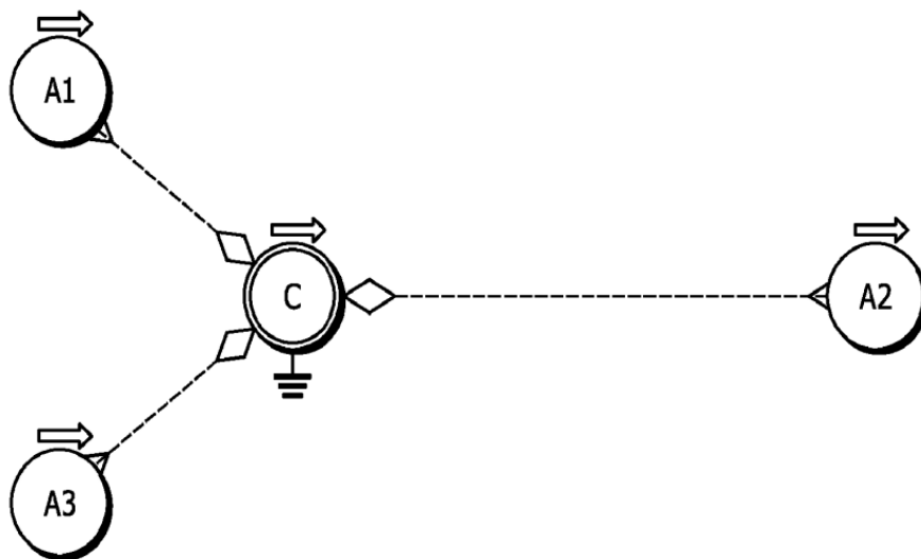


FIGURA 6

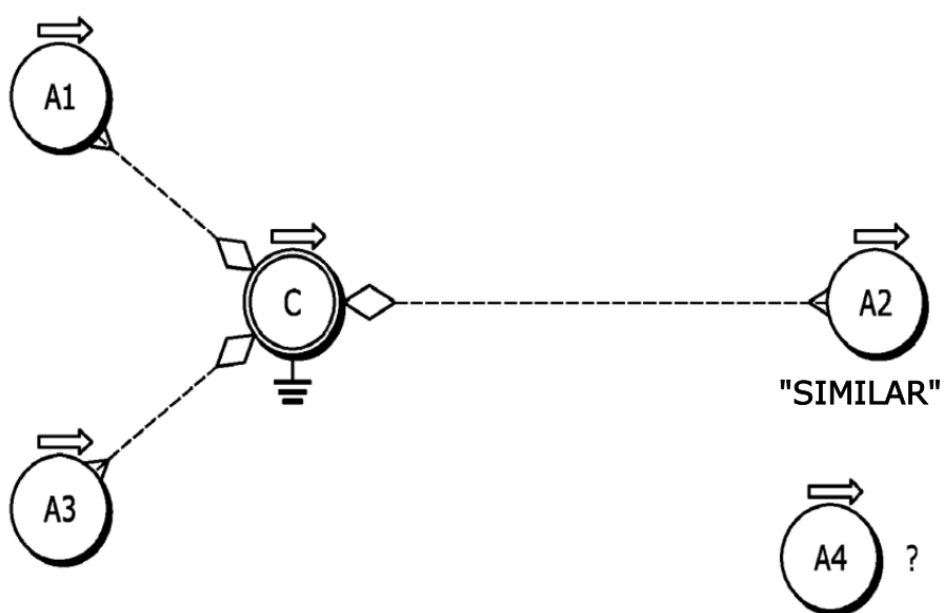


FIGURA 7

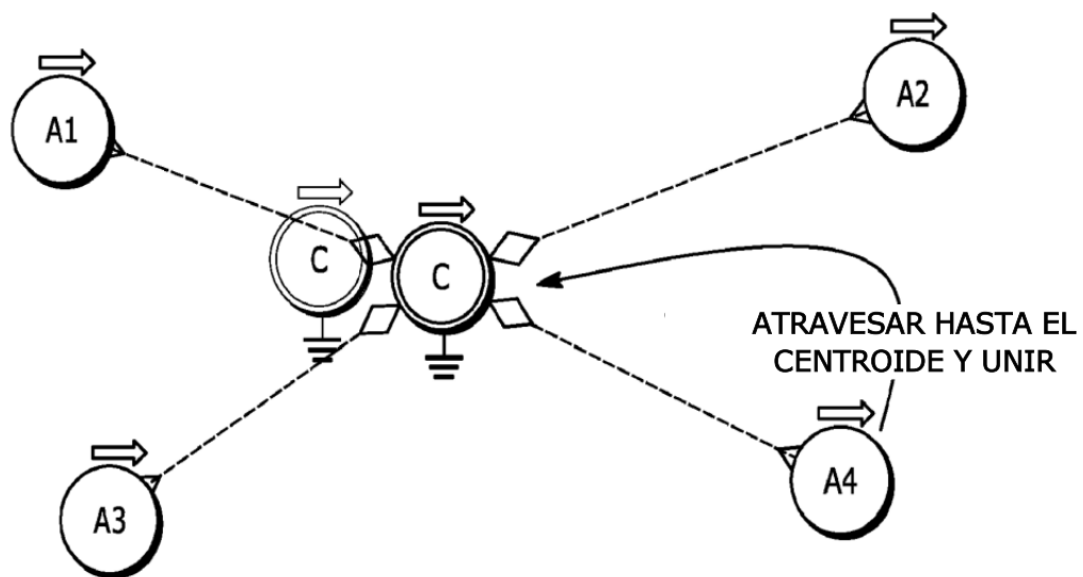


FIGURA 8

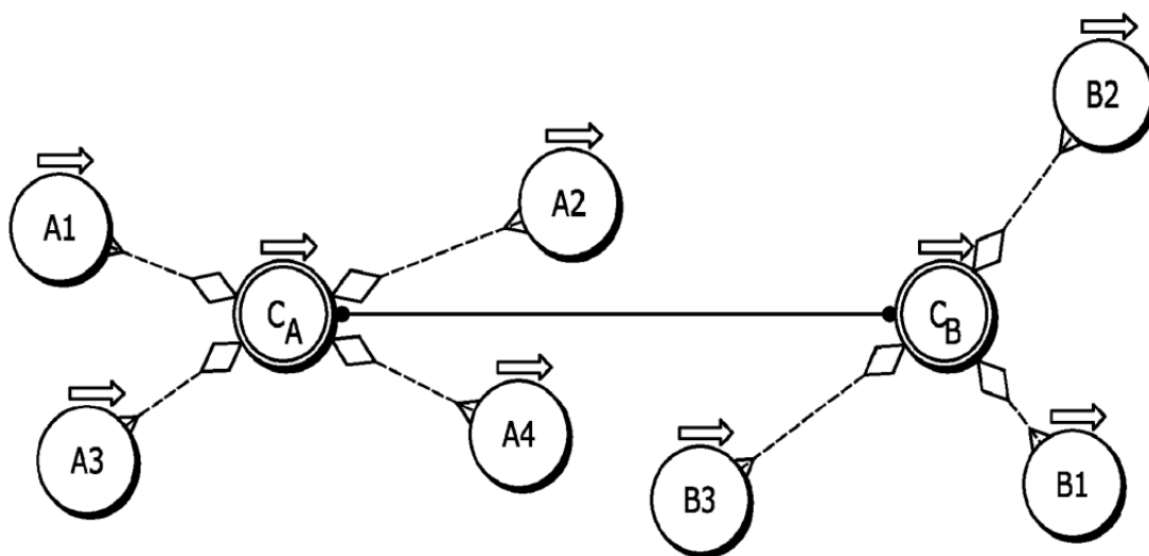


FIGURA 9

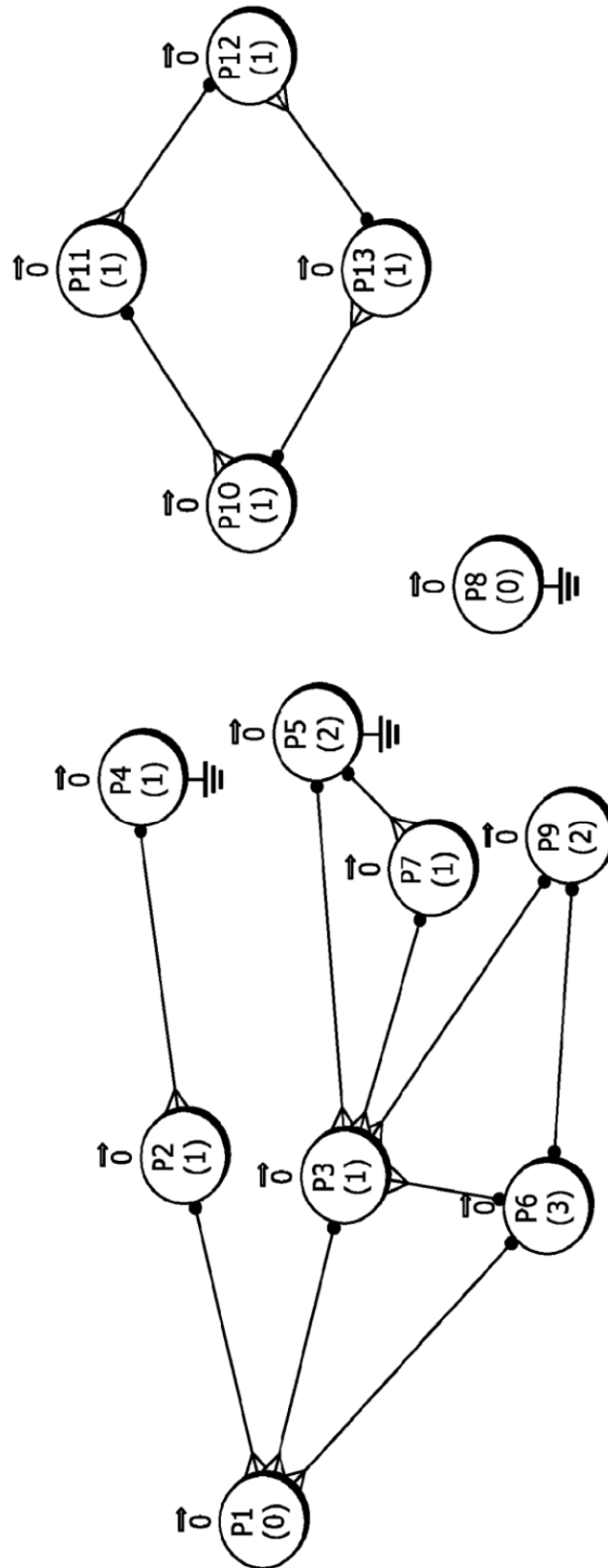


FIGURA 11

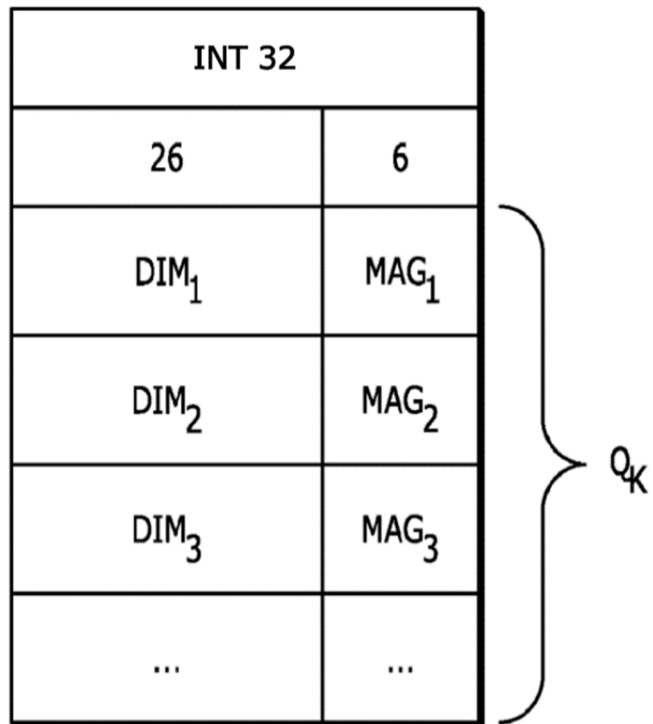


FIGURA 12

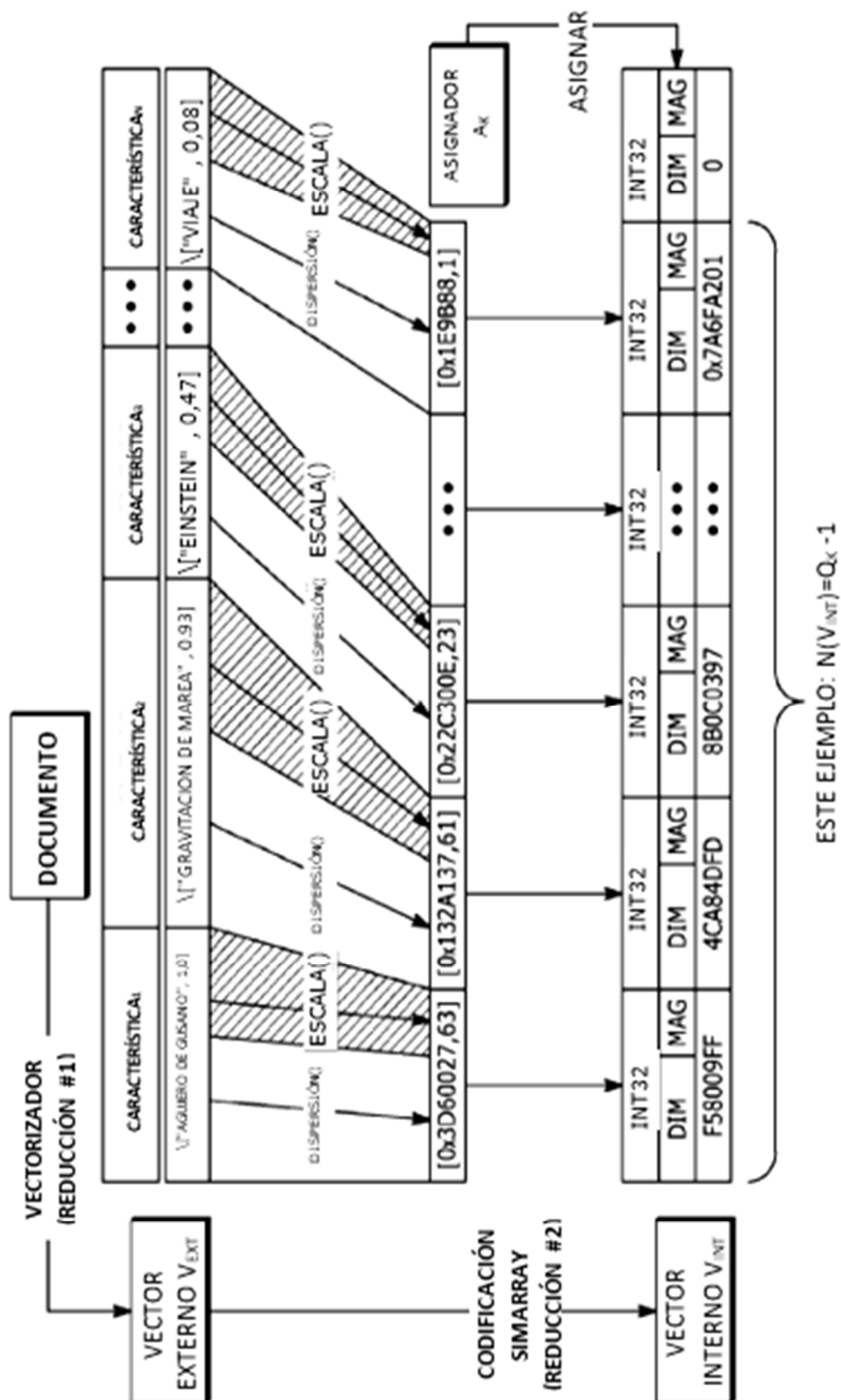


Figura 13

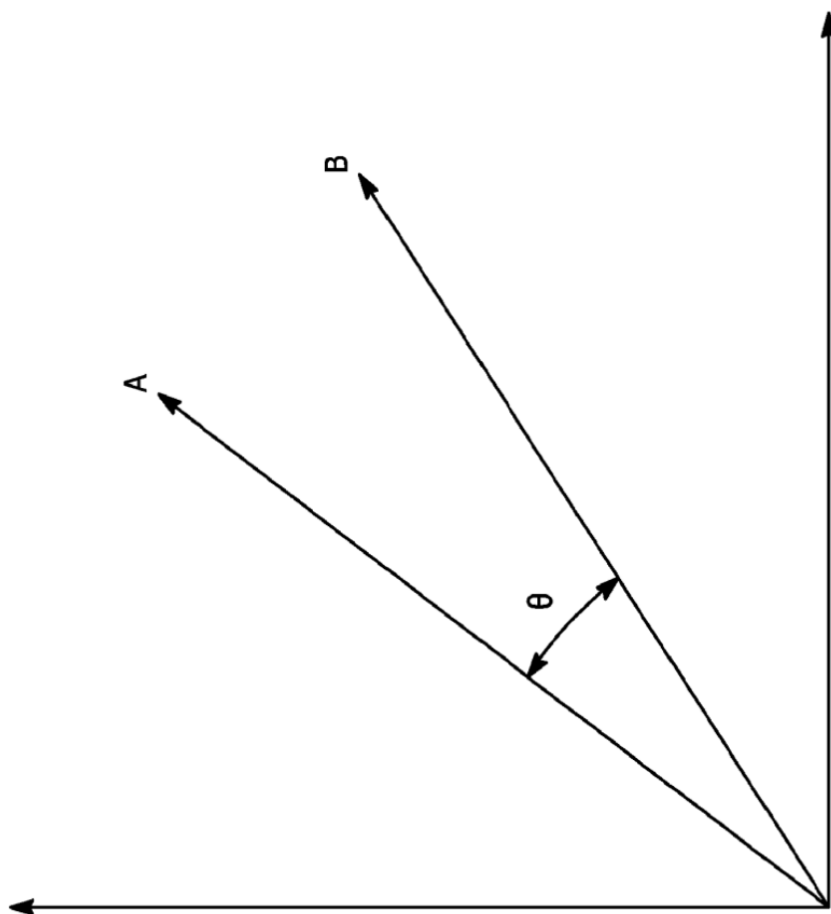


FIGURA 14

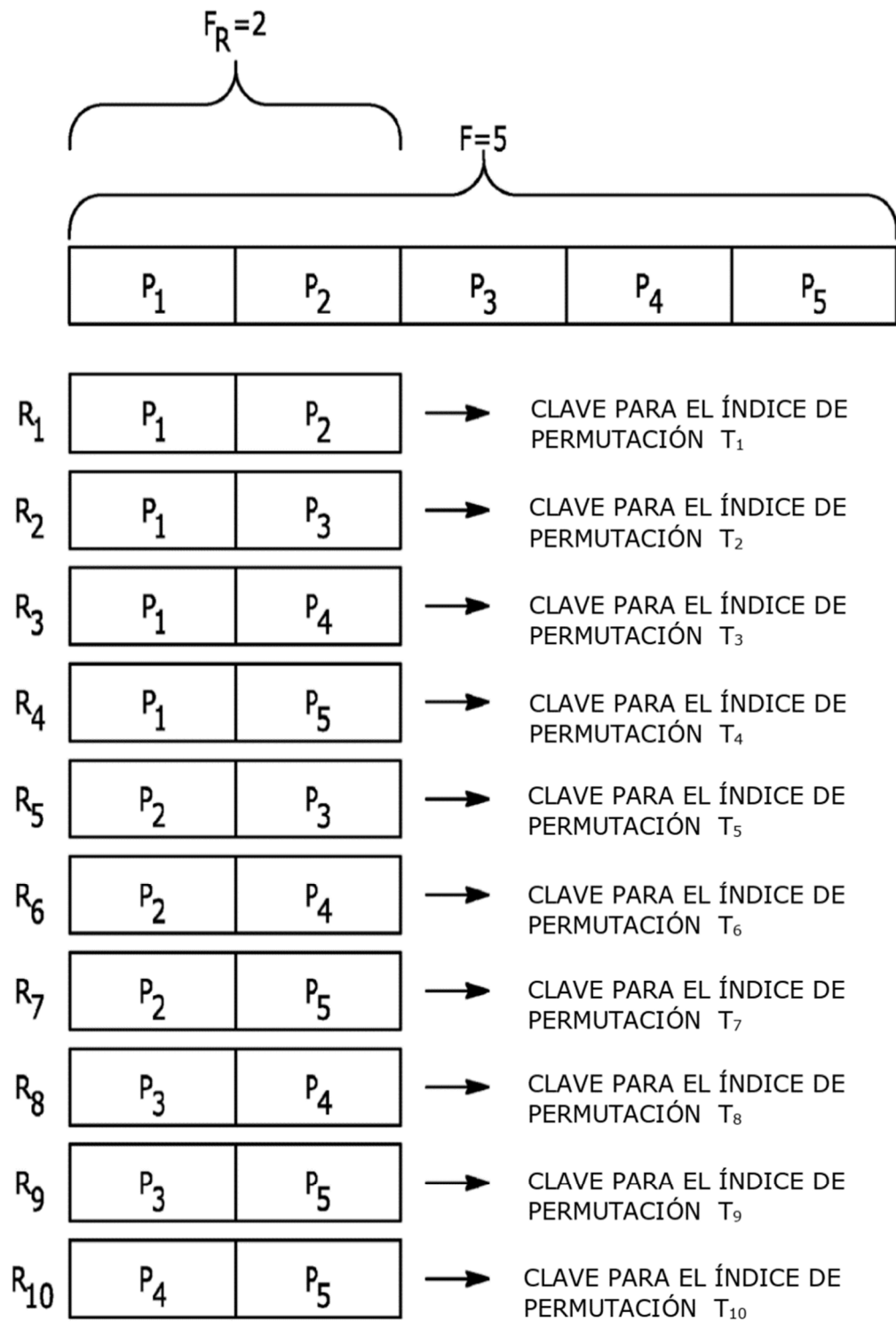


FIGURA 15

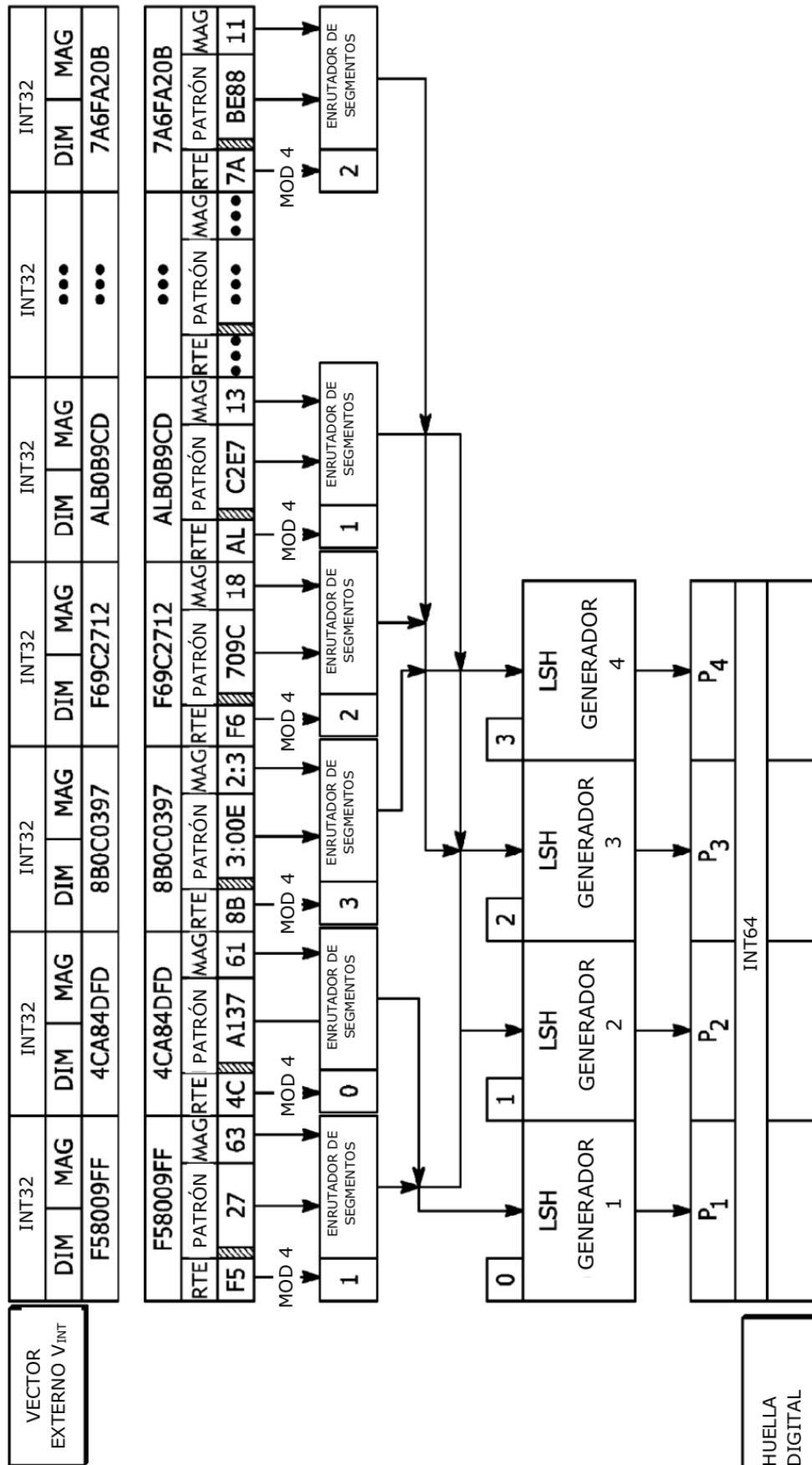


FIGURA 16

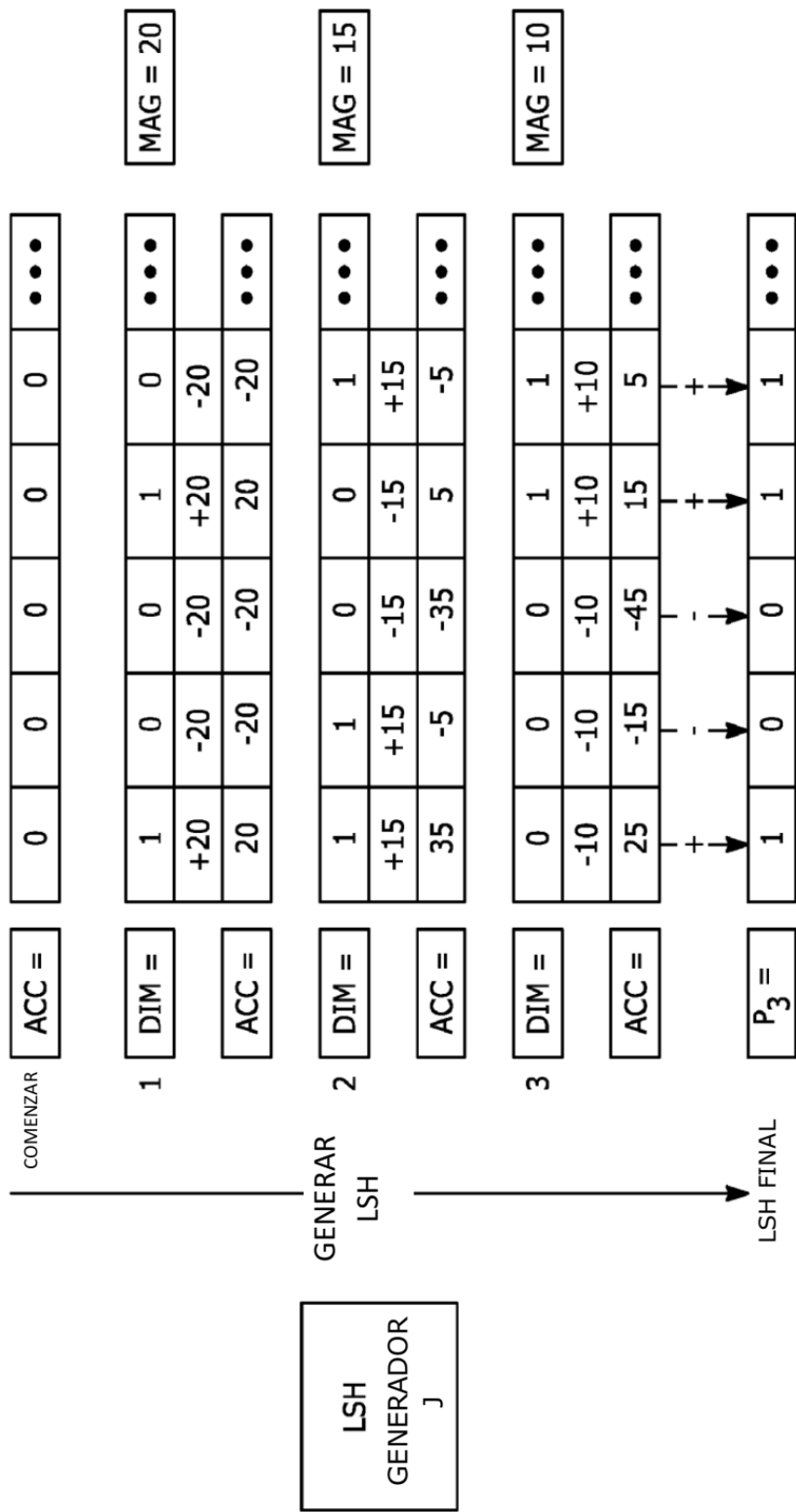


FIGURA 17

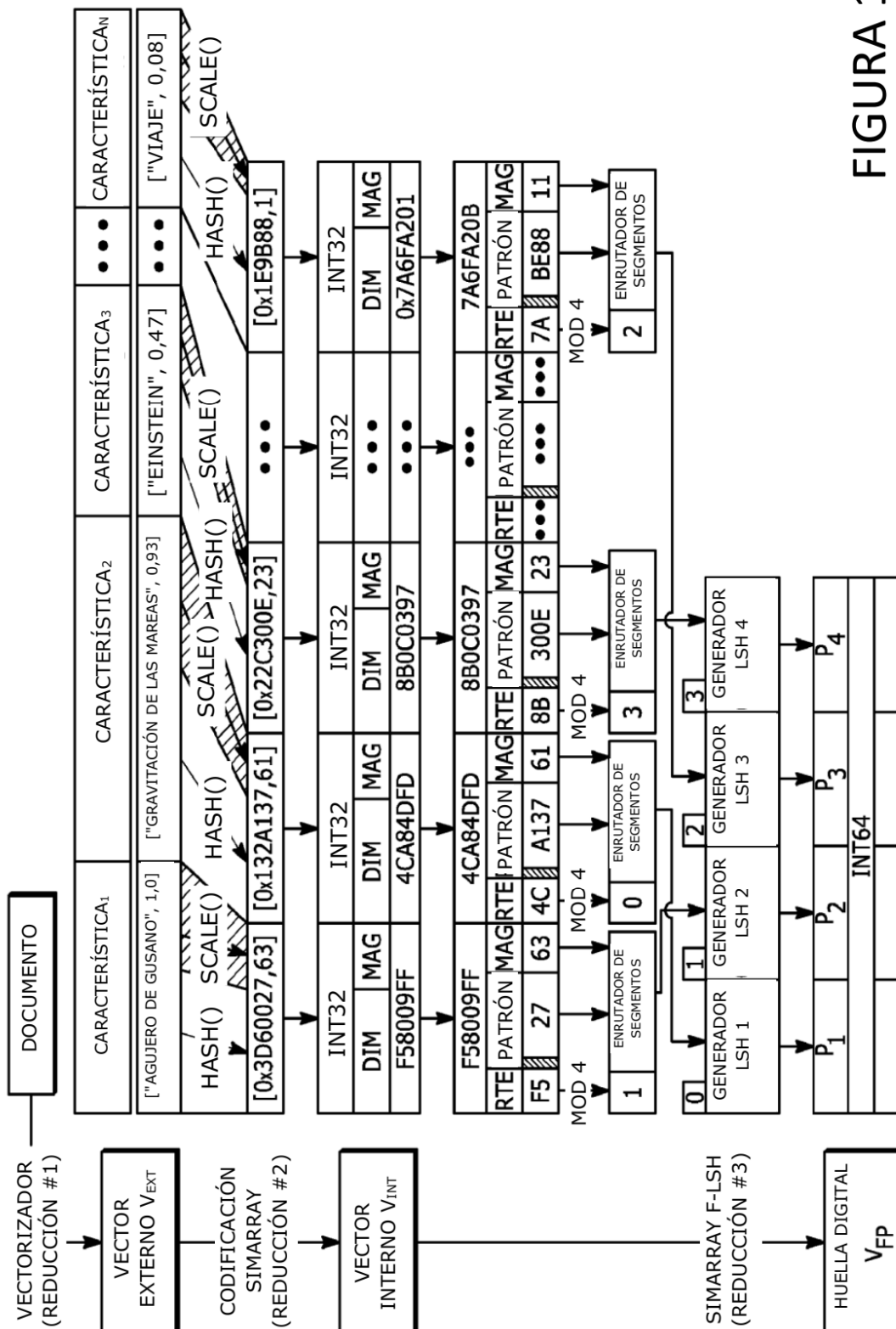


FIGURA 18

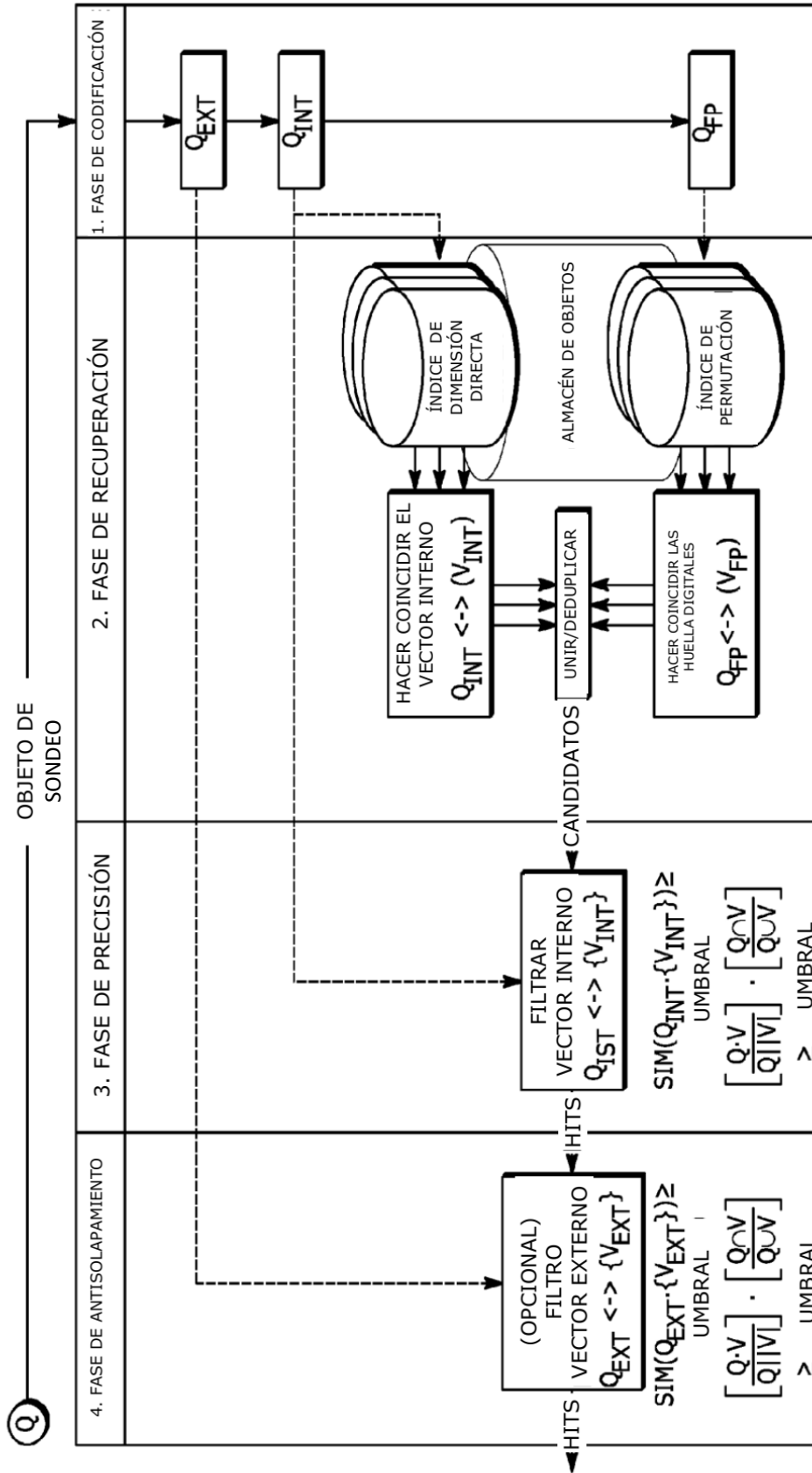


FIGURA 19

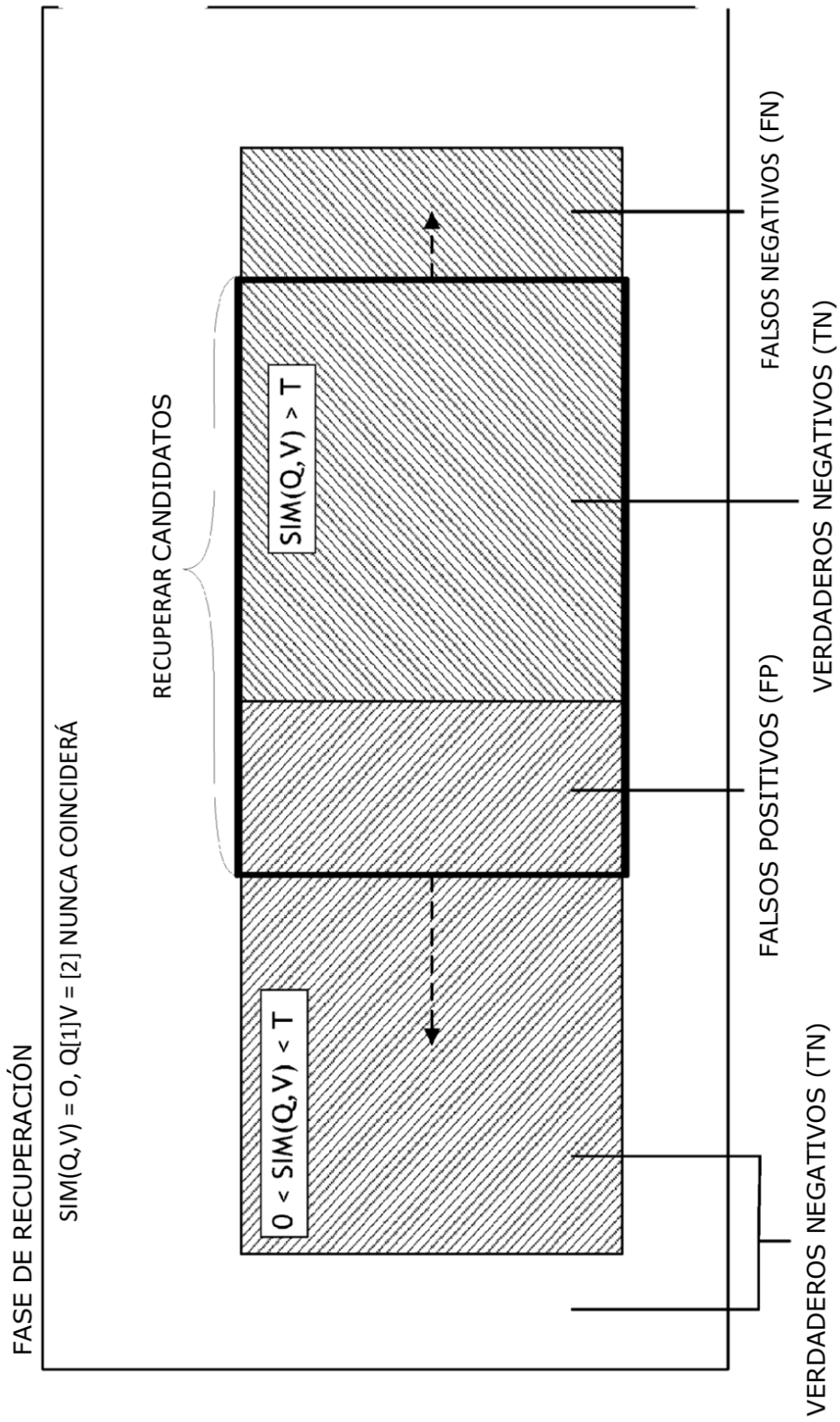


FIGURA 20

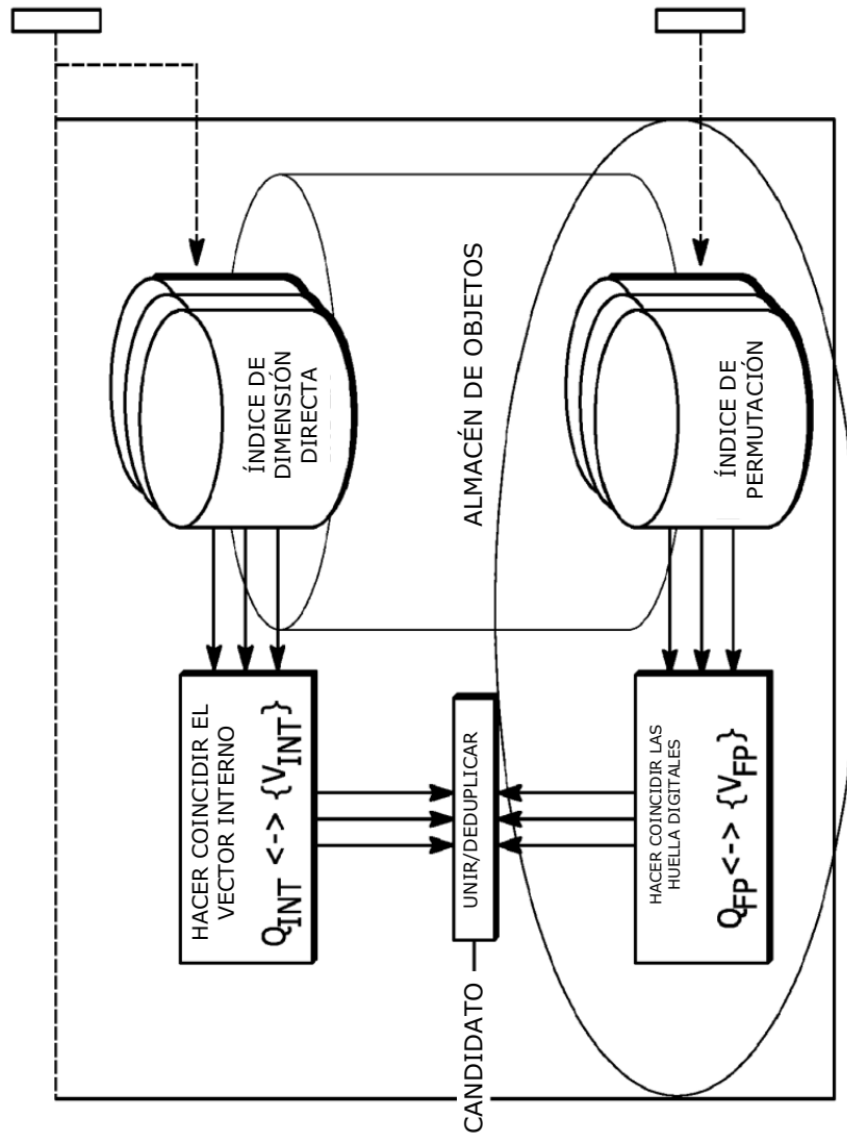


FIGURA 21

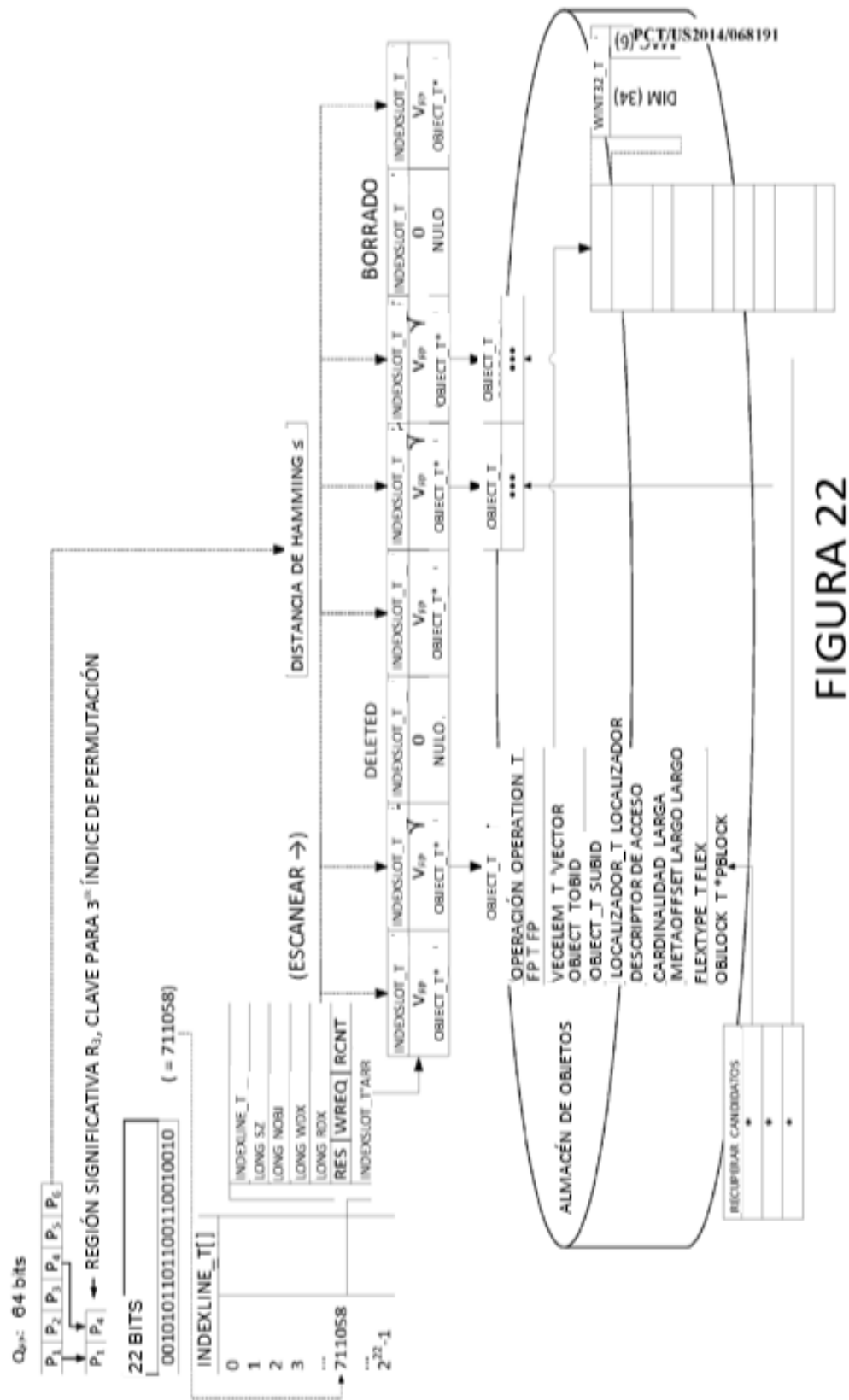


FIGURA 22

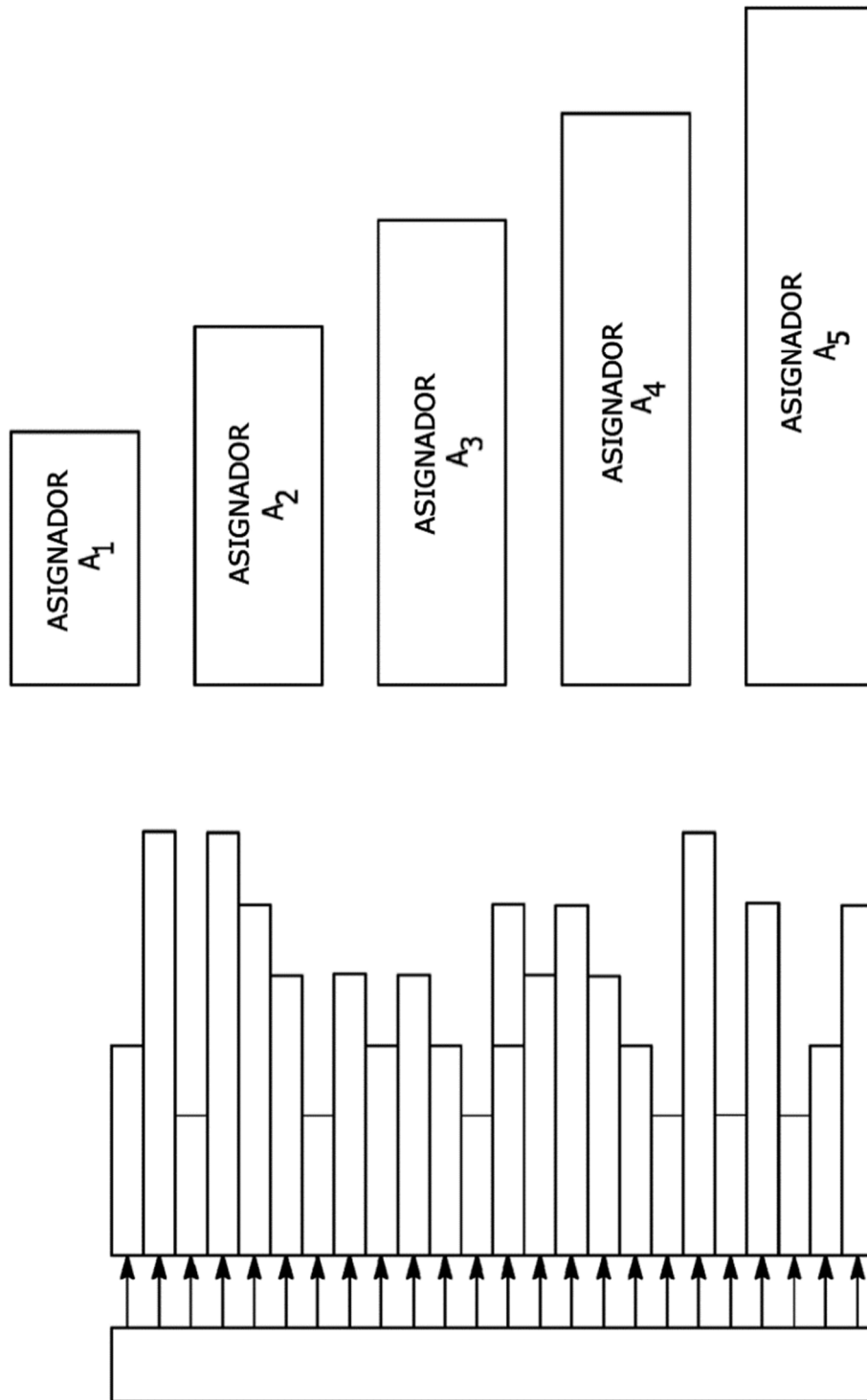


FIGURA 23

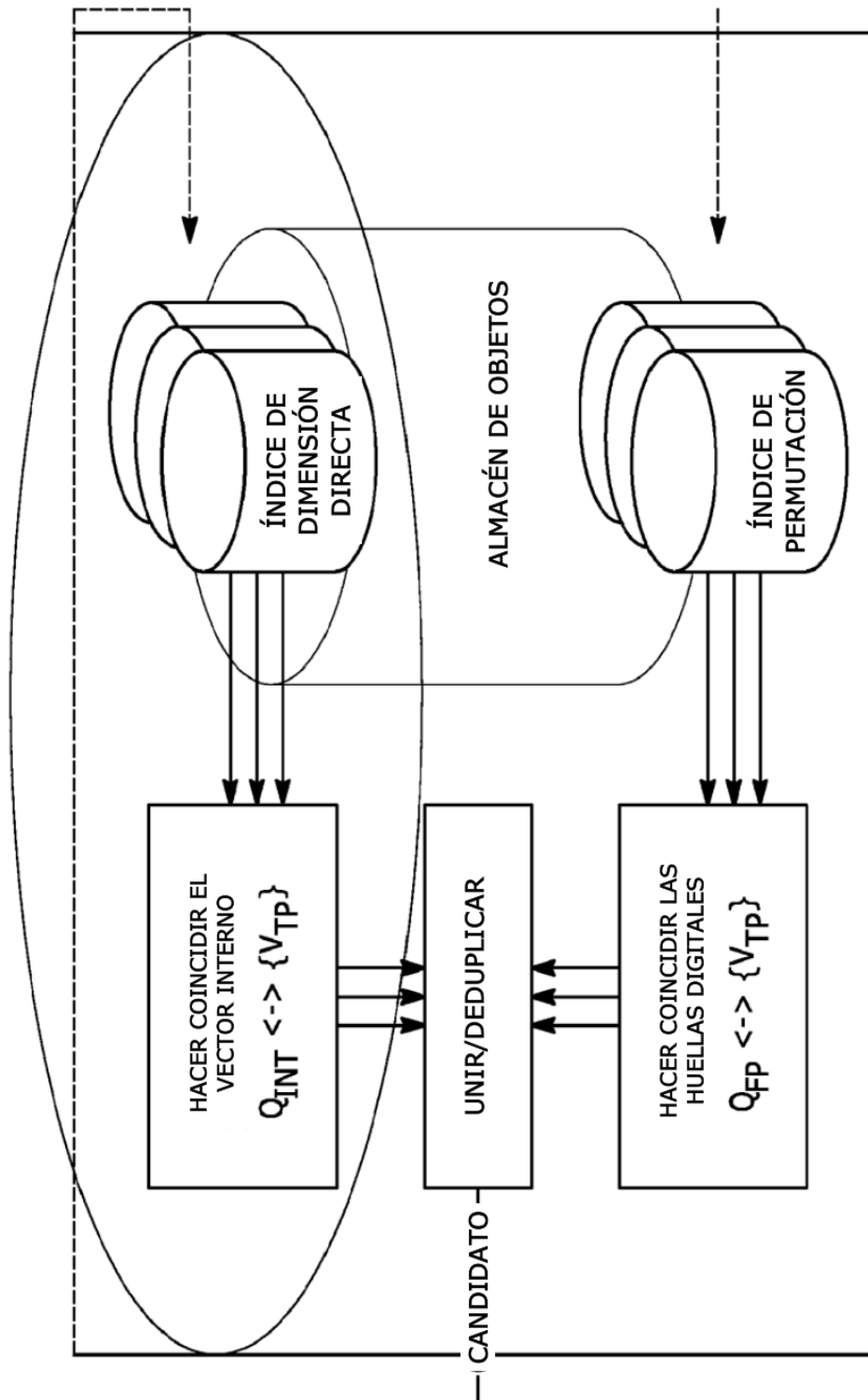


FIGURA 24

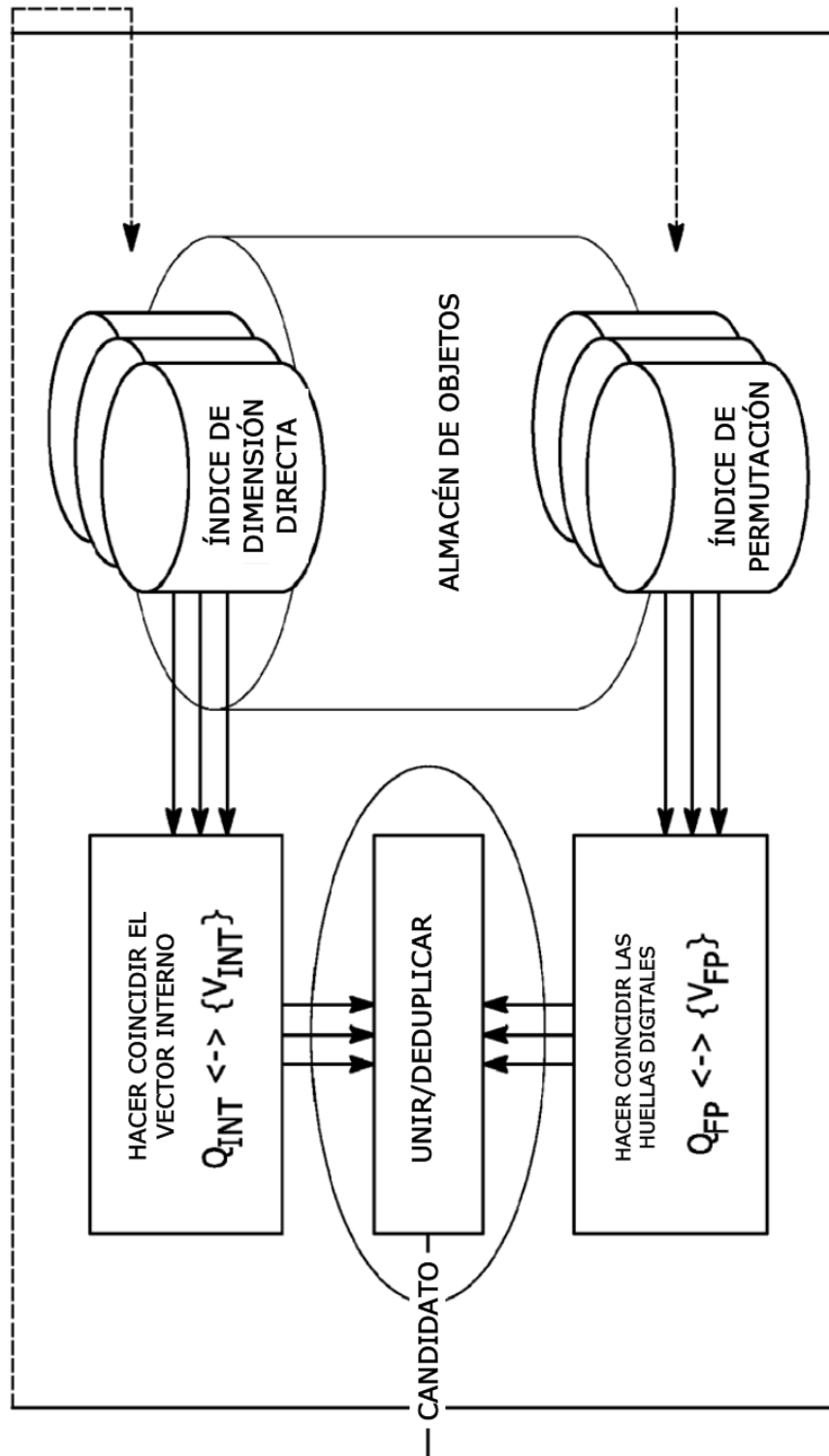


FIGURA 25

FASE DE PRECISI3N

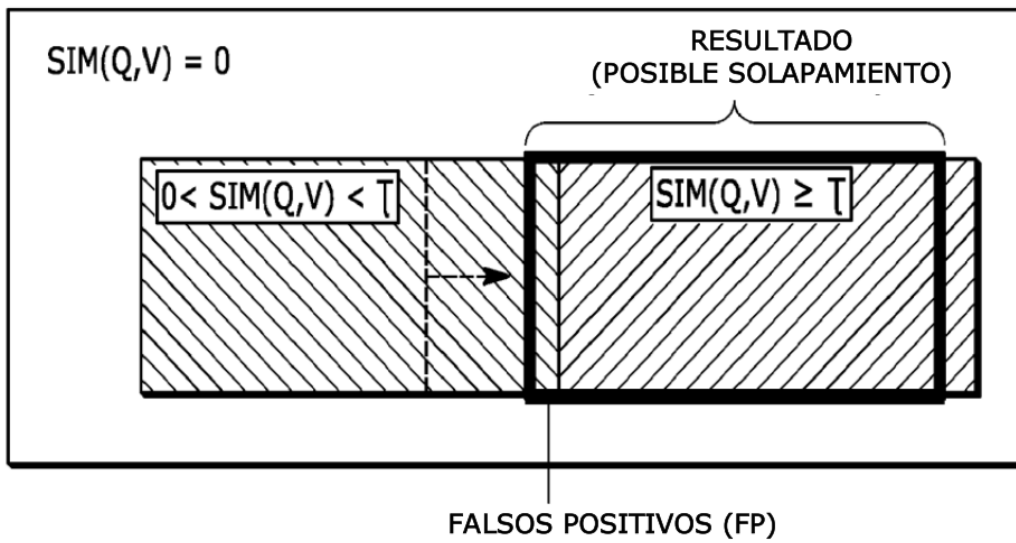


FIGURA 26

FASE ANTISOLAPAMIENTO

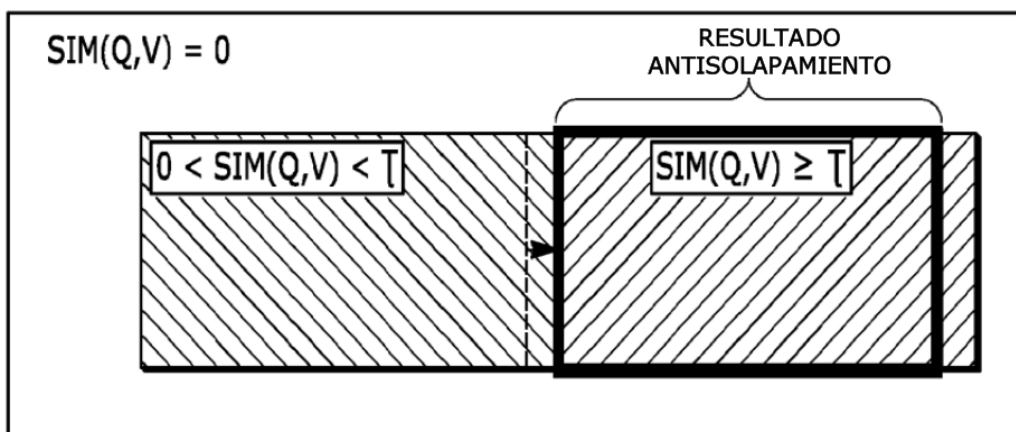


FIGURA 27

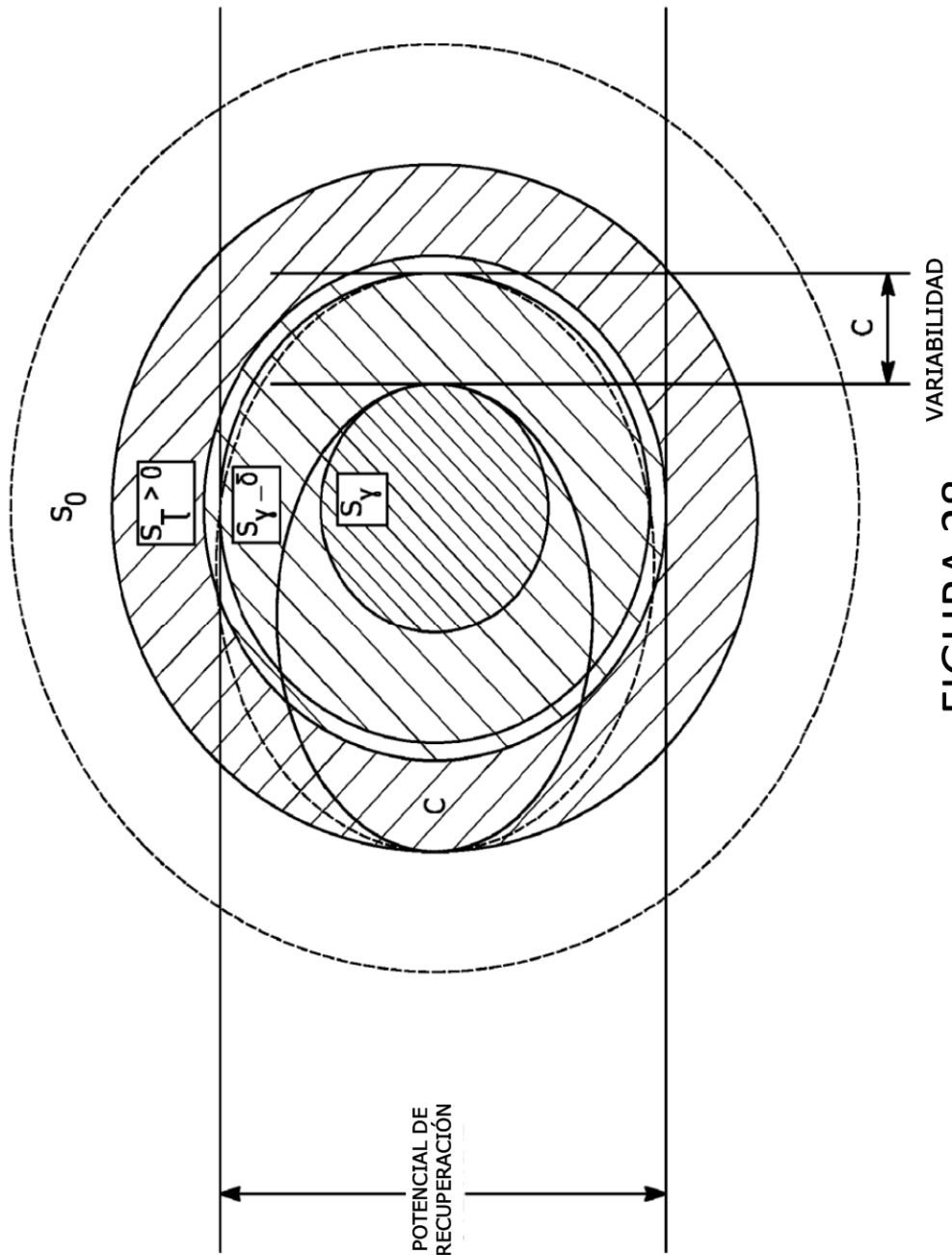


FIGURA 28

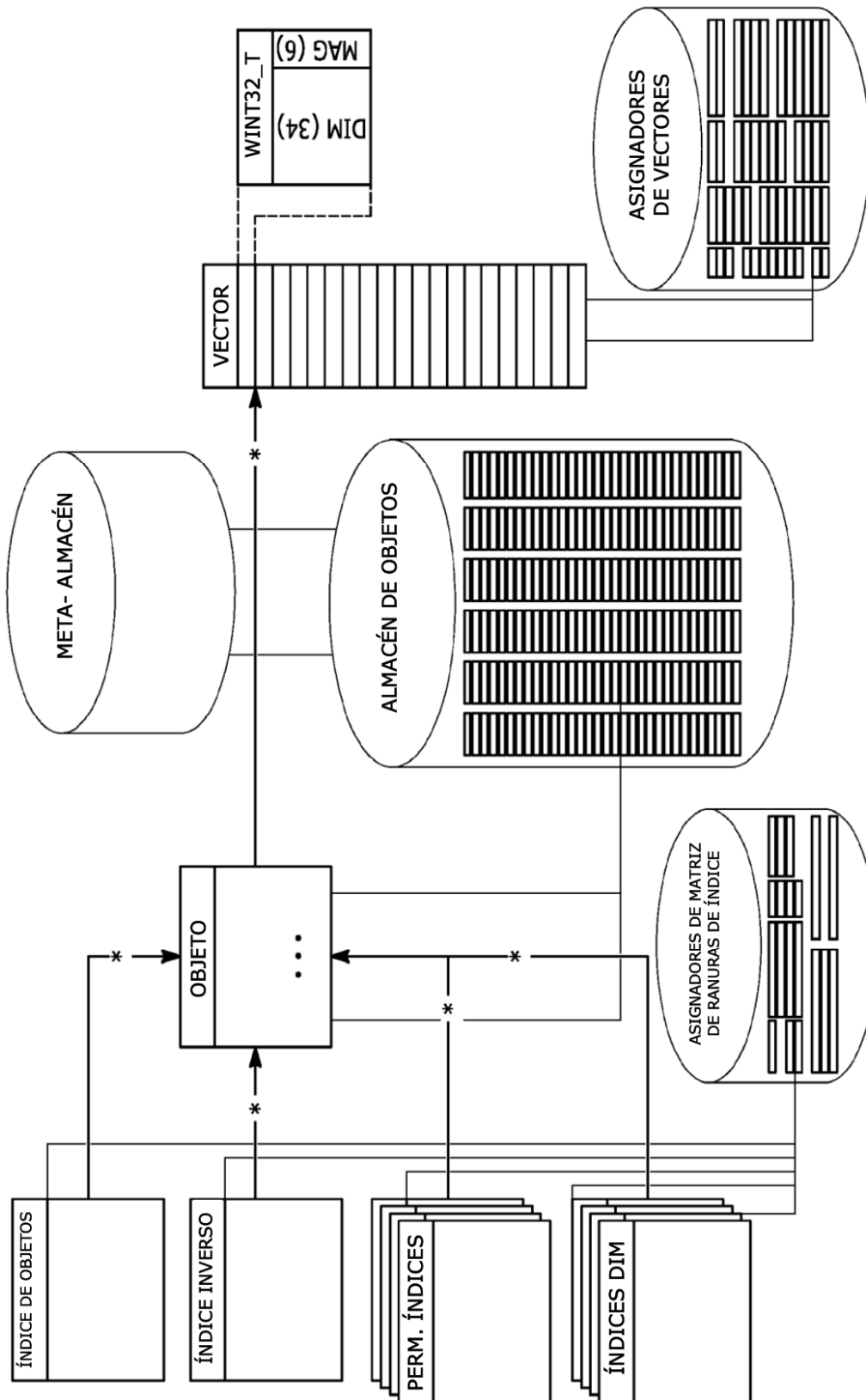


FIGURA 29

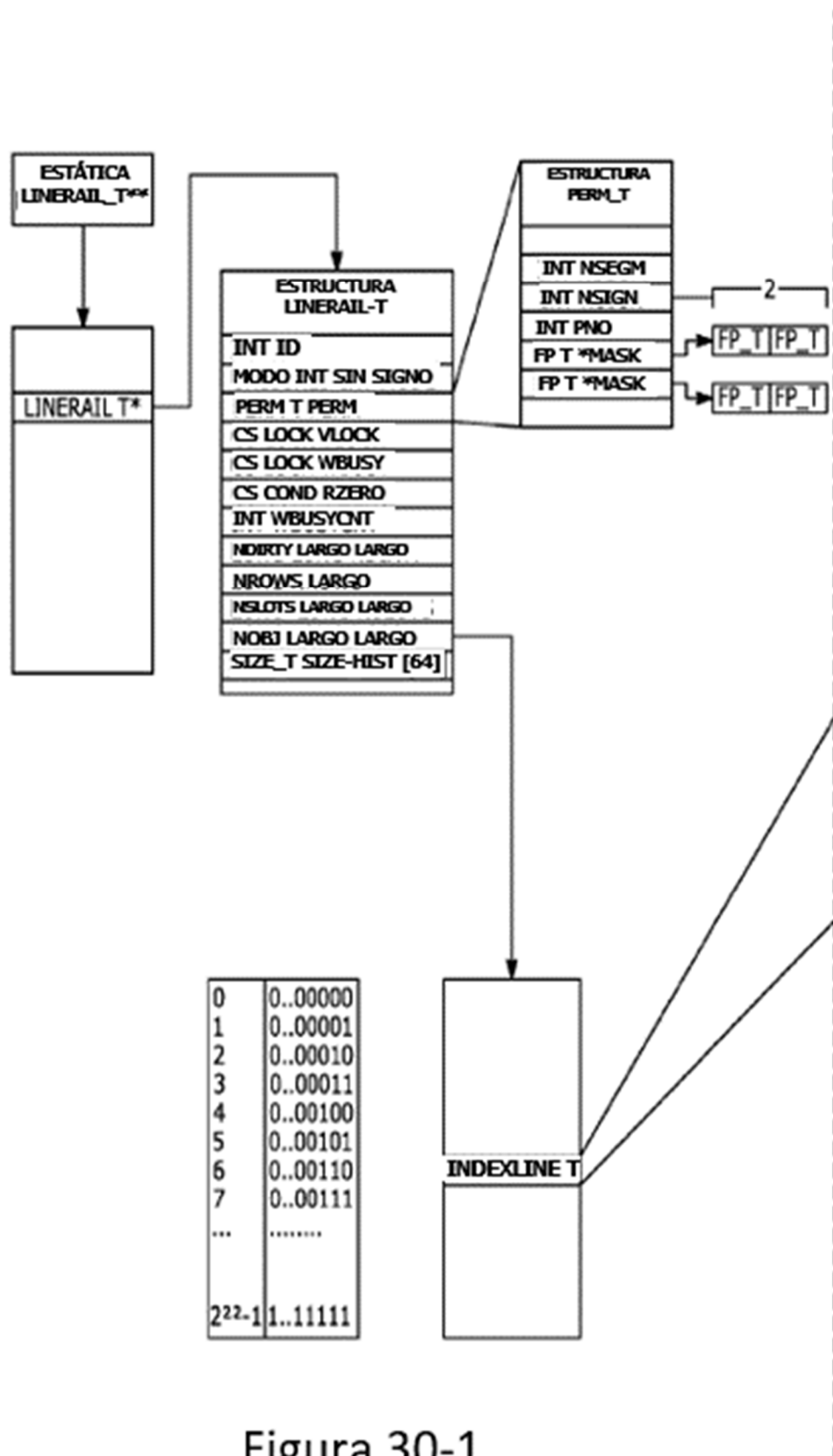


Figura 30-1

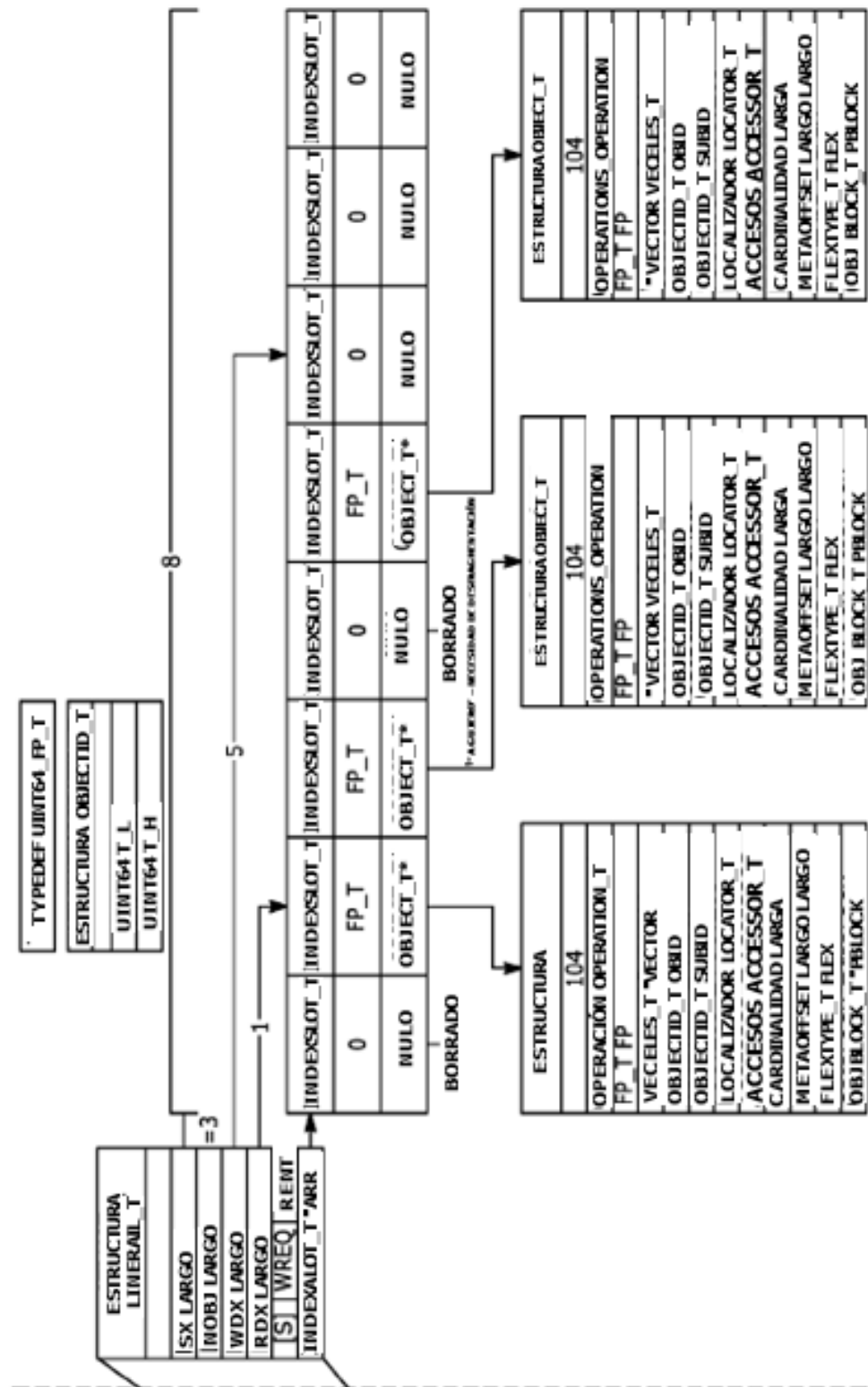


Figura 30-2

INDEXHEADER T									
HT		ENCABEZADO II			ENCABEZADO III				
PARANS		MASK			ASIGNADORES				

ENCABEZADO PARTE I (PARÁMETROS)						
UNIT64_T	UNIT64_T	UNIT64_T	UNIT64_T	UNIT64_T	UNIT64_T	
API_VER	ID:	PNO	OPCNT	NSEGM	NSIGN	
0x100000007	0,1,2....	0,1,2....	99 411 451	6	2	

ENCABEZADO PARTE II (MÁSCARAS FP PERMUTADAS/DESPLAZAMIENTOS)							
UNIT64_T	UNIT64_T	UNIT64_T	UNIT64_T	UNIT64_T	UNIT64_T	UNIT64_T	UNIT64_T
MASK_0	MASK_1	MASK_2	MASK_3	MASK_4	MASK_5	MASK_6	MASK_7
0x67FF	0x3FF800	0x0	0x0	0x0	0x0	0x0	0x0

ENCABEZADO PARTE III (INFORMACIÓN DEL ASIGNADOR)						
UNIT64_T	UNIT64_T	UNIT64_T	UNIT64_T	UNIT64_T	UNIT64_T	
NBUCKETS	NOVERSEX	NOVERSLOTES	AIDX=0	AIDX=1	AIDX=2	
100	5	671 744	998,643	2 496 202	307,276	

REGISTROS									
INDEXLINE 0				INDEXLINE 1			INDEXLINE 2		
LONG LONG	IDXRECBUF_T			IDXRECBUF_T			LONG LONG	LONG LONG	
NOBJ	OBJREF			OBJREF			NOBJ	NOBJ	
2	REGISTRO	BLOQUE	COMPENSAR	REGISTRO	BLOQUE	COMPENSAR	0	1	

FIGURA 31-1

					REGISTRO				
					NROWS				

UNIT64_T	UNIT64_T	UNIT64_T	UNIT64_T
NROWS	NSLOTS	NOBJ	MOD0
4 114,304	61 102,264	35 651,791	0X1 (NORMAL)

UNIT64_T	UNIT64_T	UNIT64_T	UNIT64_T	UNIT64_T	UNIT64_T	UNIT64_T	UNIT64_T
SHIFT_0	SHIFT_1	SHIFT_2	SHIFT_3	SHIFT_4	SHIFT_5	SHIFT_6	SHIFT_7
0	0	0	0	0	0	0	0

UNIT64_T	UNIT64_T	UNIT64_T
AIDX=3	...	AIDX=99
15,123	...	0

		...	LÍNEA DE ÍNDICE 4.194.303						
IDXRECBUF_T		...	LONG LONG	IDXRECBUF_T			IDXRECBUF_T		
OBJREF		...	NOBJ	OBJREF			OBJREF		
REG.	BLOQUE	COMPENSAR	...	2	REG.	BLOQUE	COMPENSAR	REG.	BLOQUE

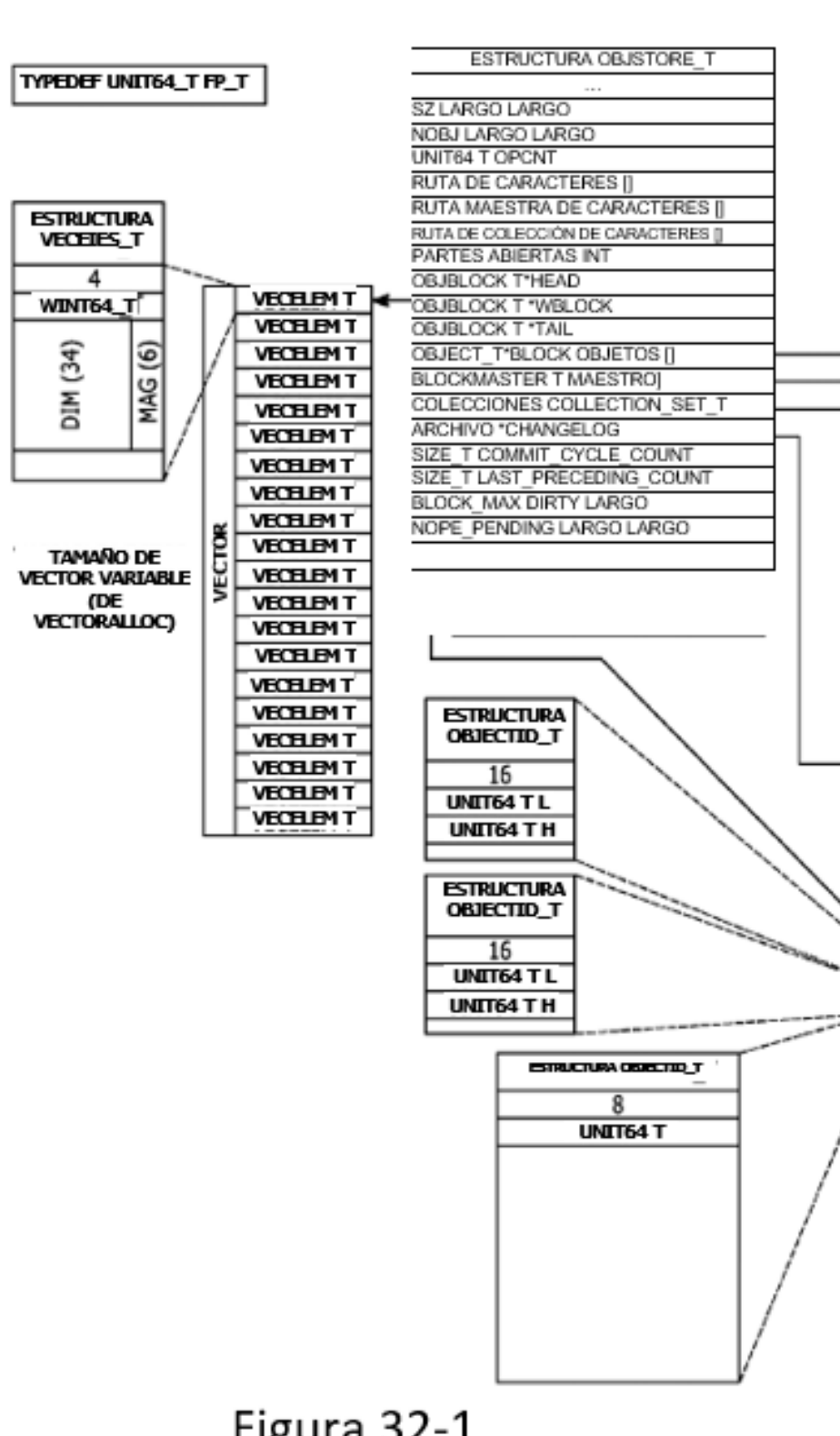


Figura 32-1

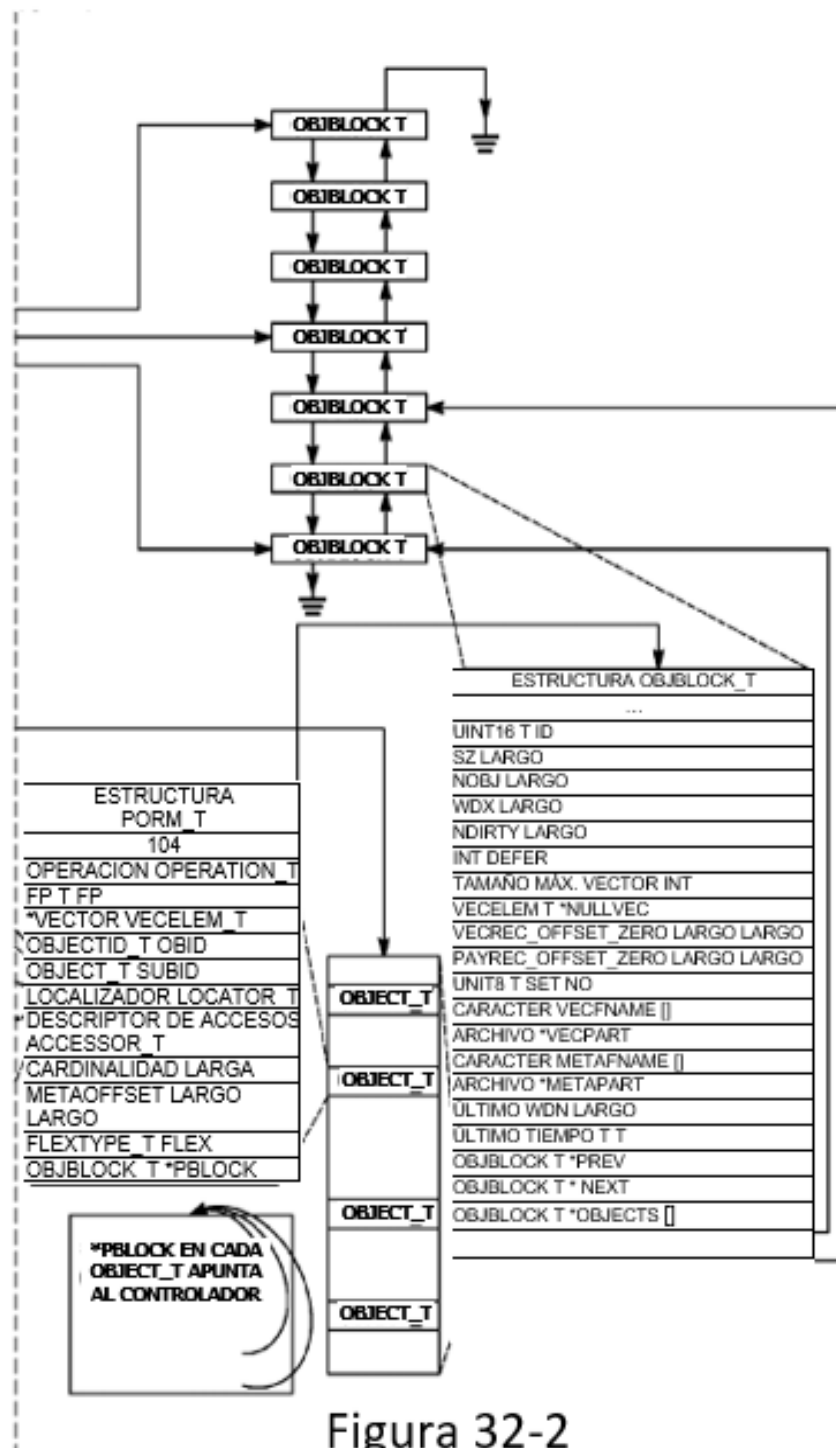


Figura 32-2

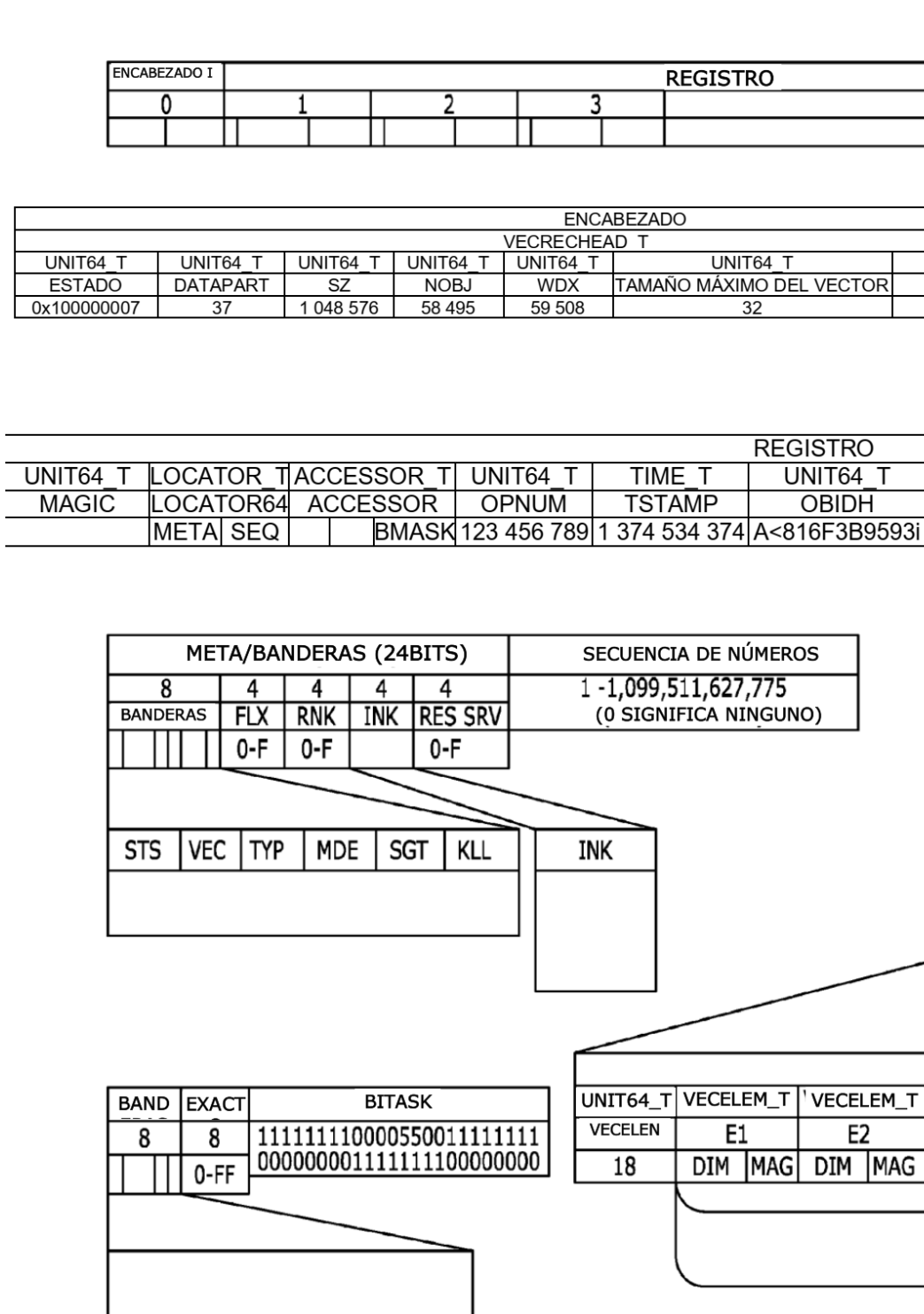


FIGURA 33-1

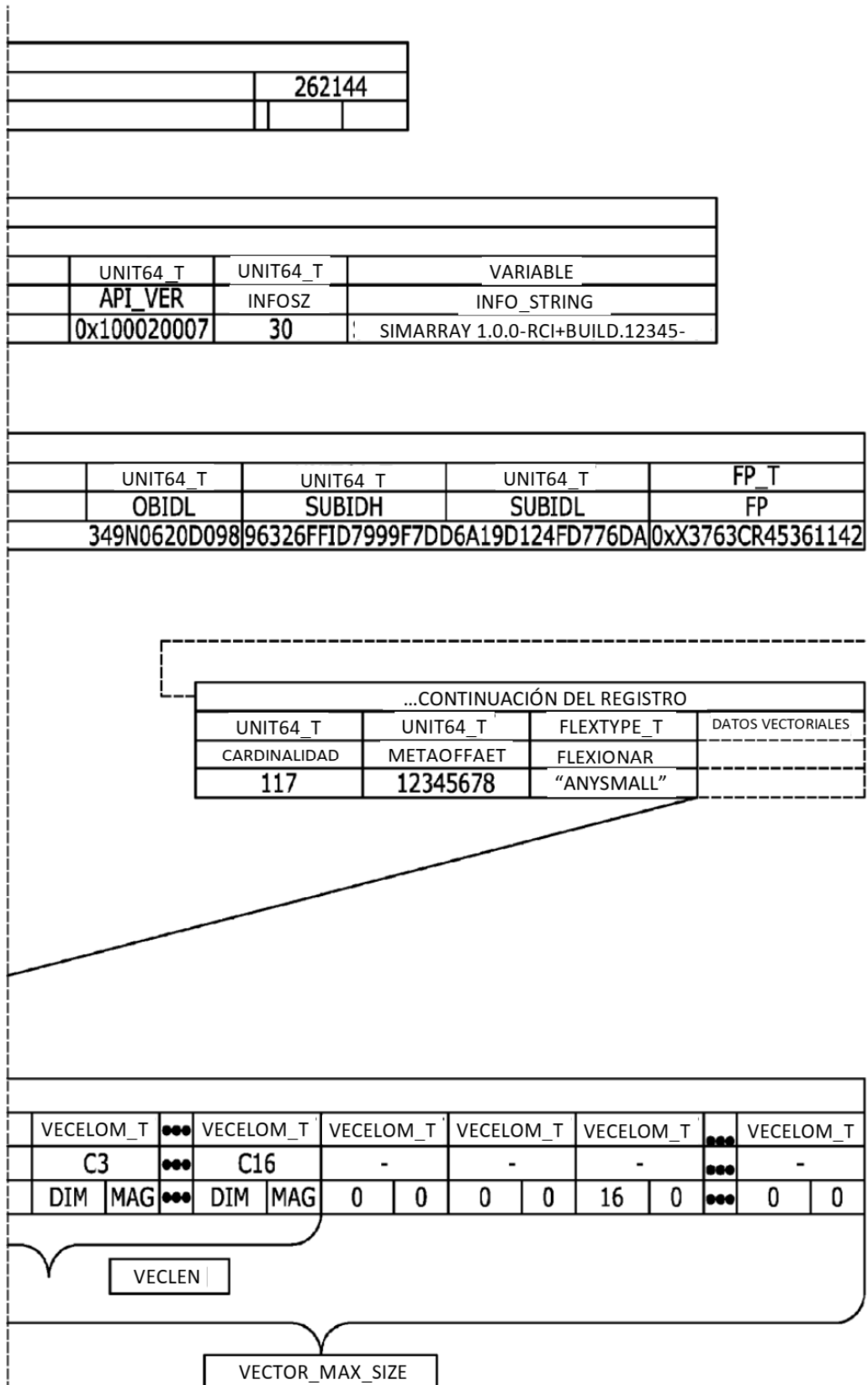


FIGURA 33-2

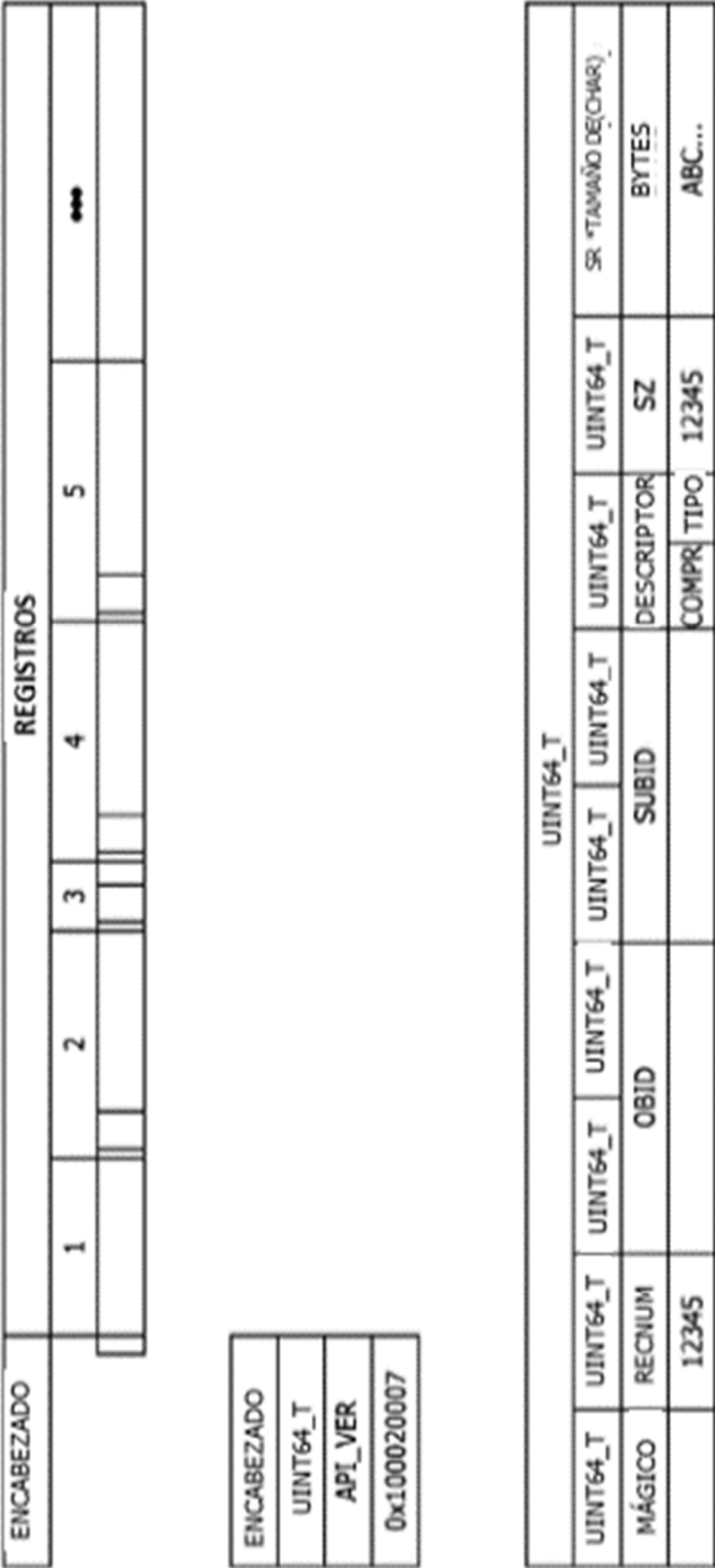


Figura 34

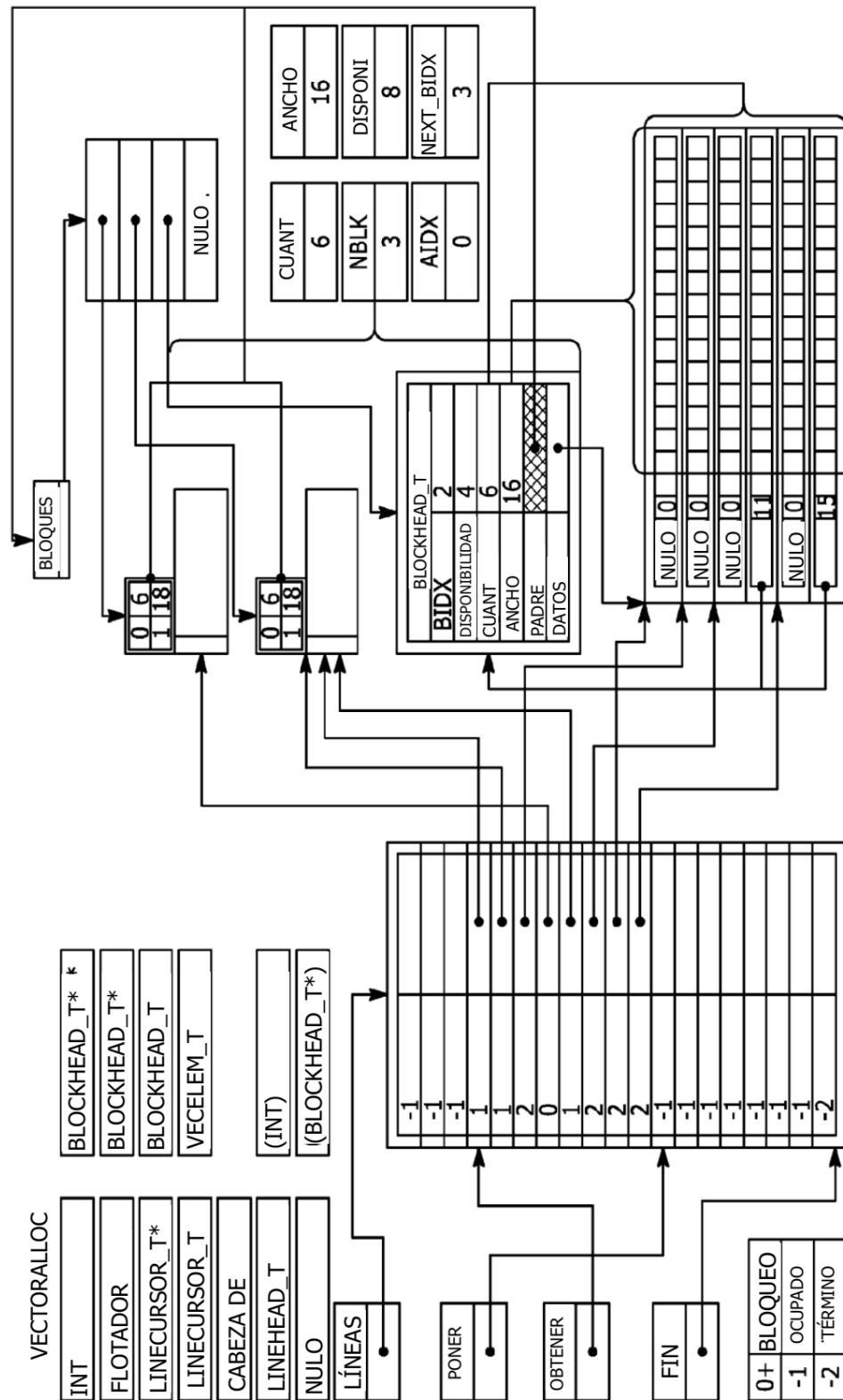


FIGURA 35

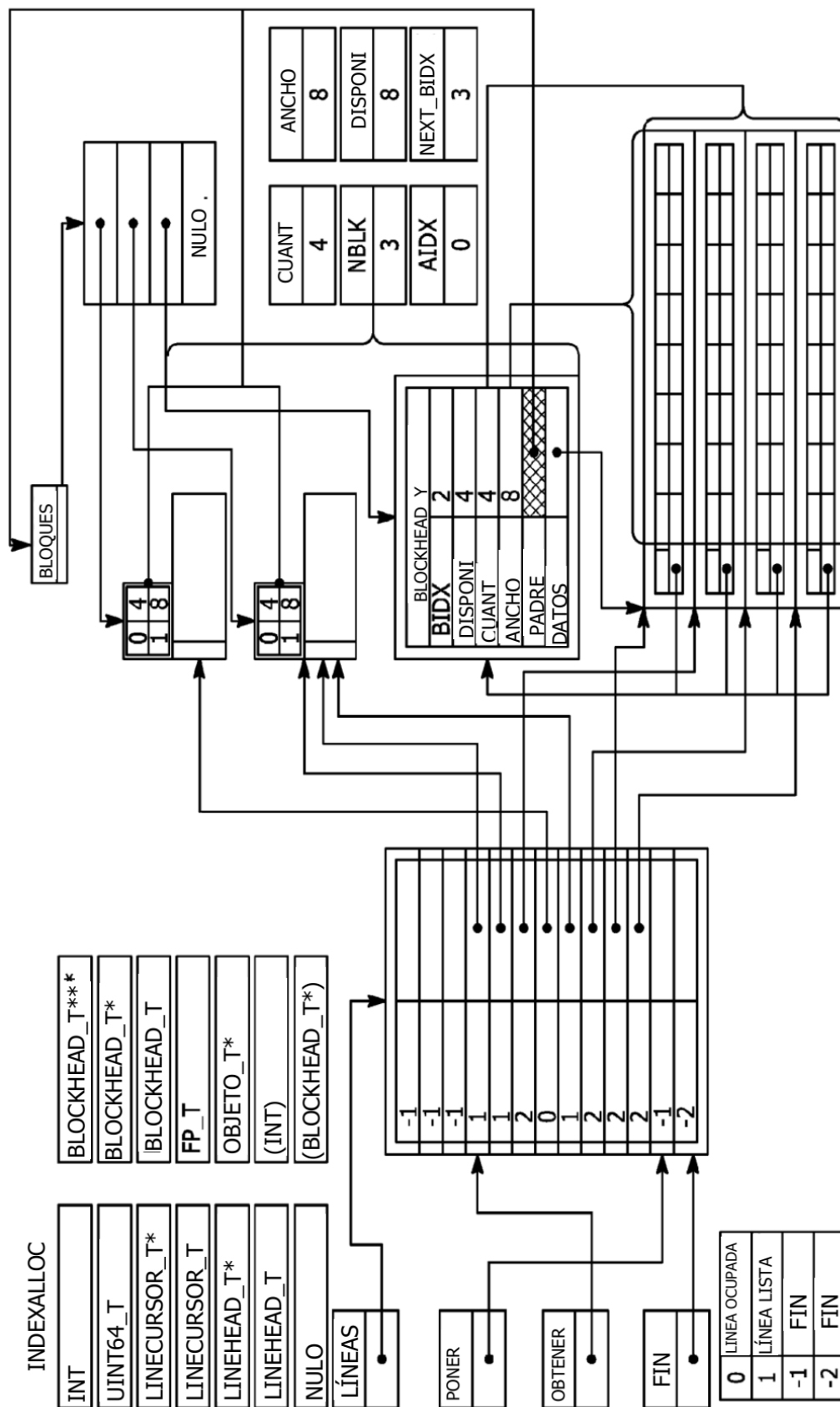


FIGURA 36

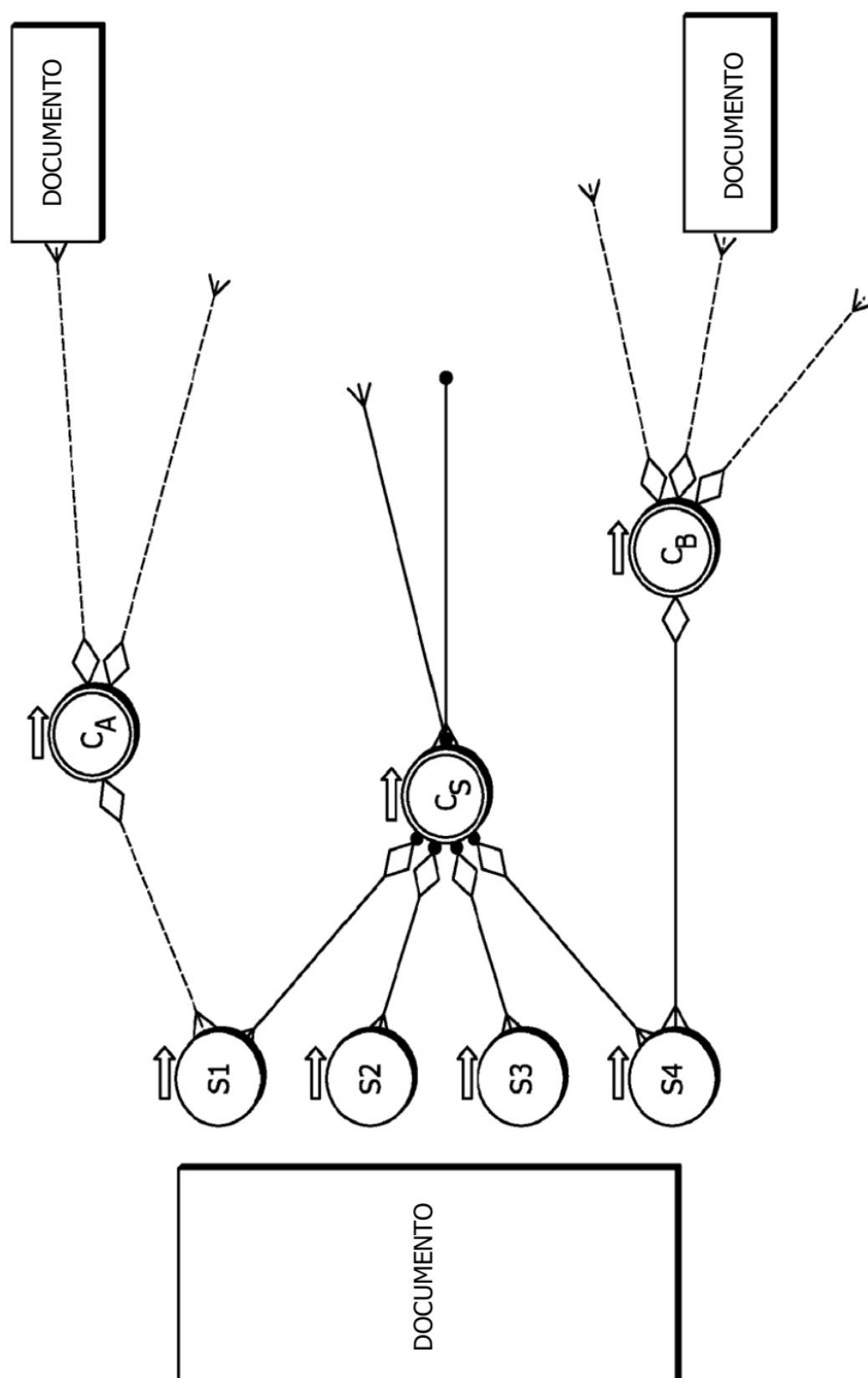


FIGURA 37

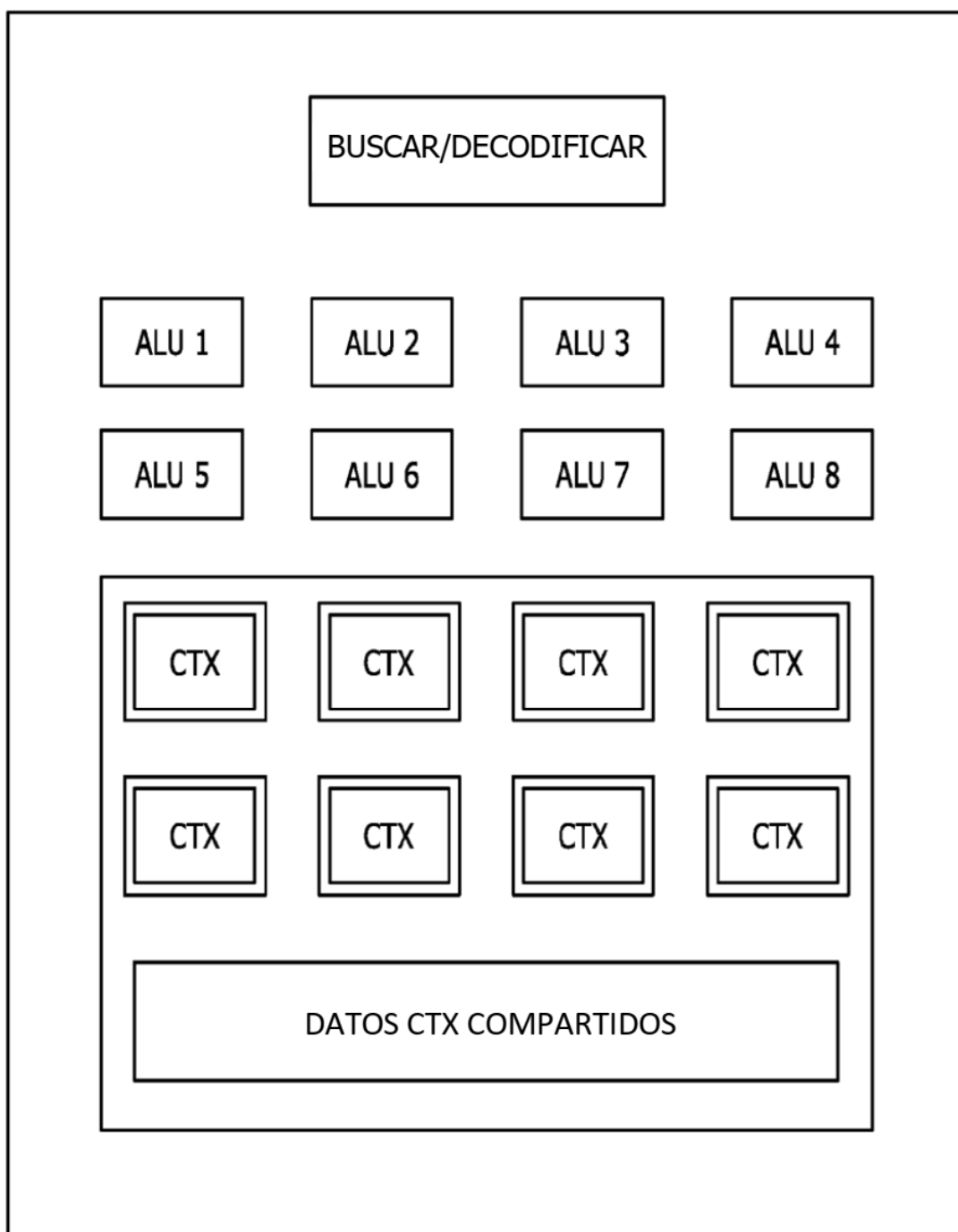


FIGURA 38

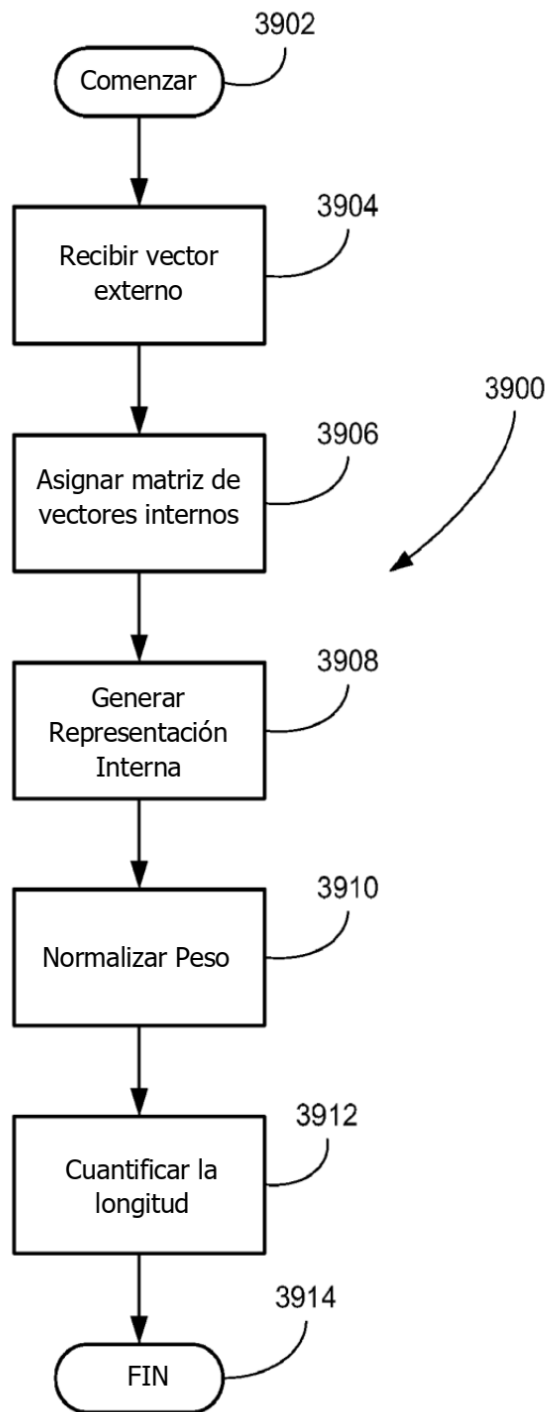


FIGURA 39

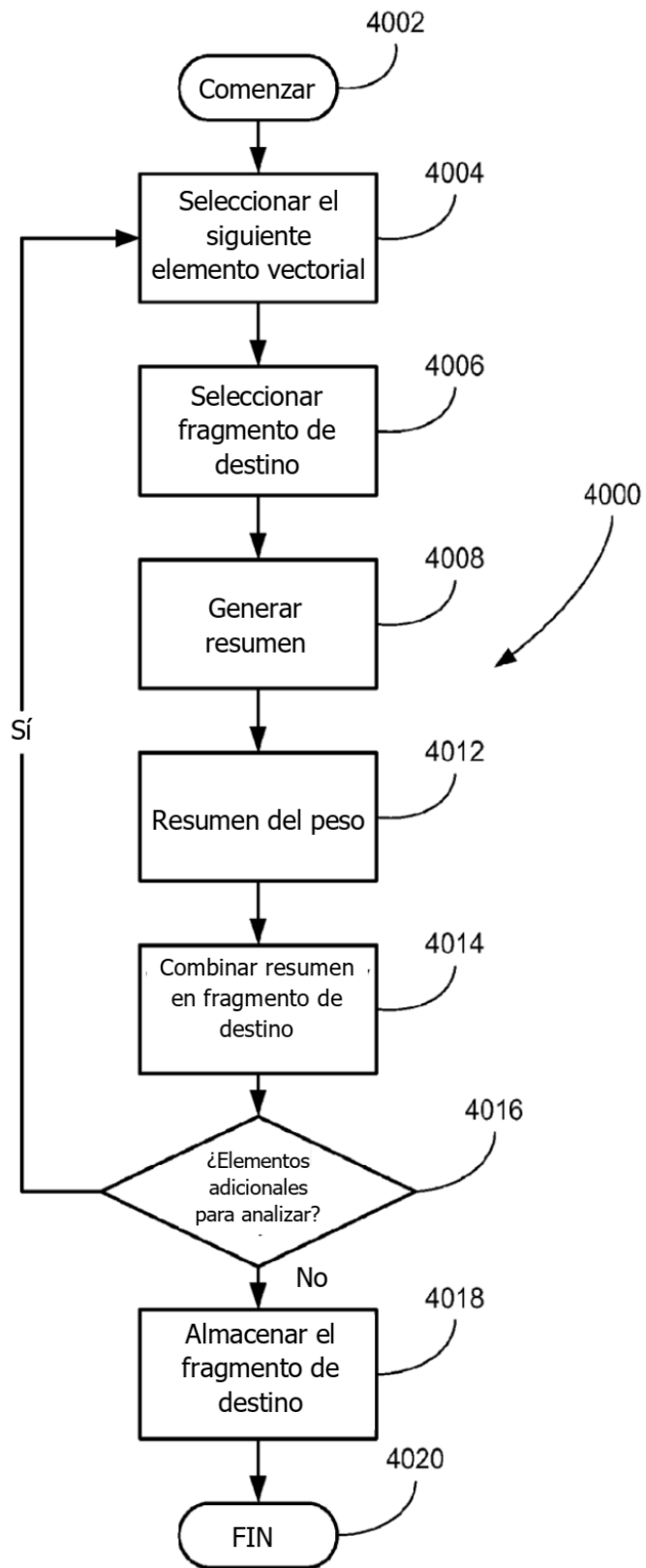


FIGURA 40

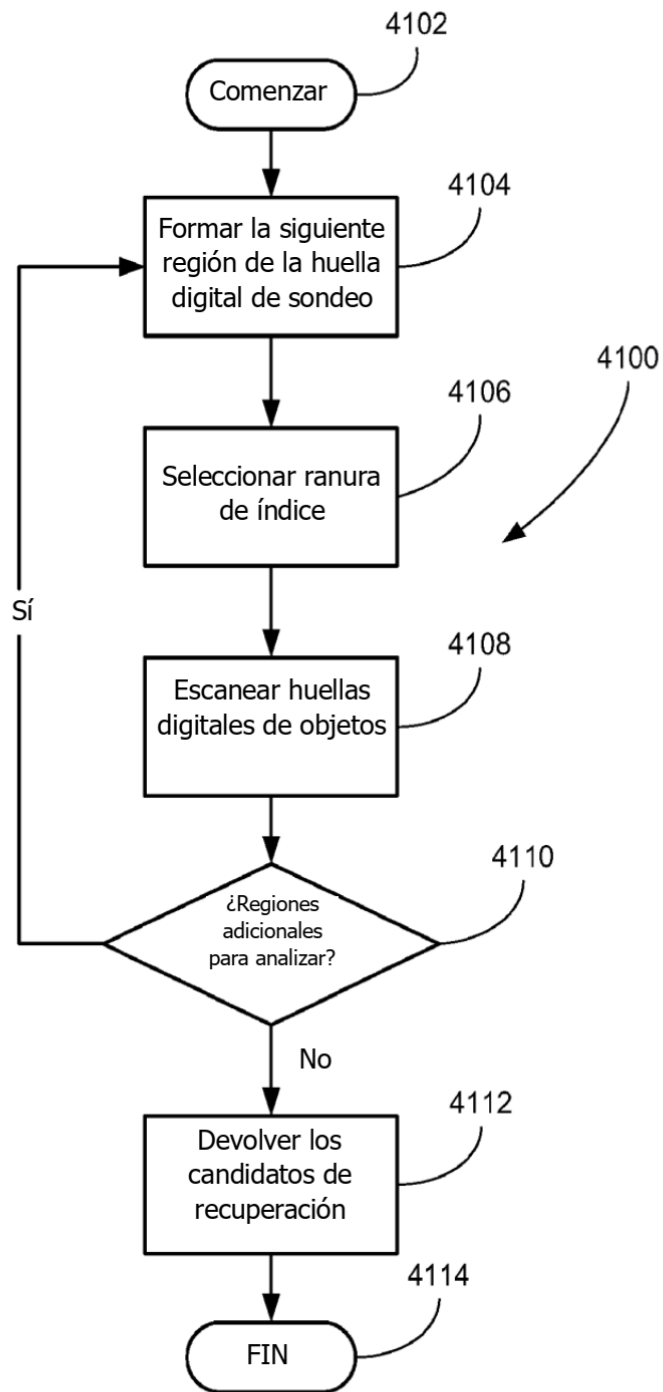


FIGURA 41

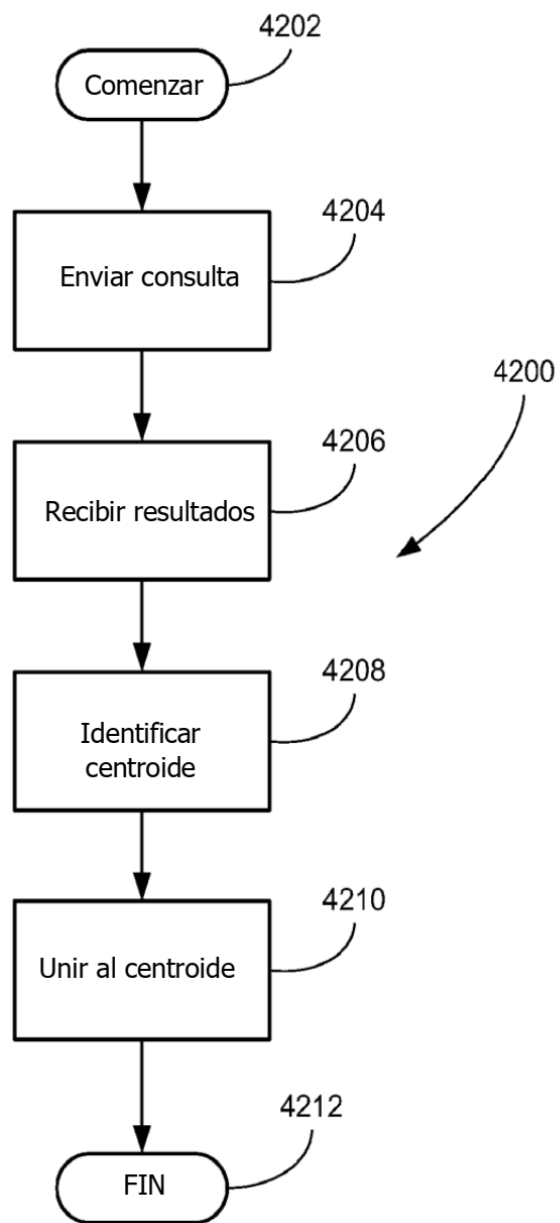


FIGURA 42