(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 553 971**

(51) Int. CI.:

G06N 5/02 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **27.04.2010 E 10717615 (8)**

(97) Fecha y número de publicación de la concesión europea: **26.08.2015 EP 2425383**

(54) Título: **Método y dispositivo para la evolución ontológica**

(30) Prioridad:

30.04.2009 EP 09159178

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
15.12.2015

(73) Titular/es:

COLLIBRA NV/SA (50.0%)
Oorlogskruisenlaan 116
1120 Brussel, BE y
VRIJE UNIVERSITEIT BRUSSEL (50.0%)

(72) Inventor/es:

TROG, DAMIEN;
CHRISTIAENS, STIJN;
DE LEENHEER, PIETER GASTON MARGUERITE;
VAN DE MAELE, FELIX URBAIN YOLANDE y
MEERSMAN, ROBERT ALFONS

(74) Agente/Representante:

ILLESCAS TABOADA, Manuel

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

Método y dispositivo para la evolución ontológica

Campo de la invención

La presente invención se refiere, en general, a la ingeniería de ontología. Más específicamente, la presente invención se refiere a la evolución de ontología y al mantenimiento de los cambios dentro de los sistemas de ontología.

Antecedentes de la invención

Internet y otros entornos de conectividad abierta crean una fuerte demanda por compartir la semántica de datos. Los sistemas de ontología se están convirtiendo cada vez más en esenciales para casi todas las aplicaciones informáticas. Las organizaciones están mirando hacia ellos como unos recursos semánticos procesables por máquina vitales para muchas áreas de aplicación. Una ontología es un entendimiento acordado (es decir, la semántica) de un determinado dominio, axiomatizado y representado formalmente como la teoría lógica en la forma de un recurso basado en ordenador. Compartiendo una ontología, las aplicaciones autónomas y distribuidas pueden comunicarse de manera significativa para intercambiar datos y hacer de este modo transacciones interoperativas independientemente de sus tecnologías internas.

Las ontologías capturan el dominio del conocimiento de una parte específica del mundo real, por ejemplo, el conocimiento sobre la entrega del producto. Las ontologías pueden verse como una representación formal del conocimiento por un conjunto de conceptos y las relaciones entre estos conceptos dentro de un dominio. Las ontologías deben capturar este conocimiento independientemente de los requisitos de aplicación (por ejemplo, la aplicación de entrega de producto de cliente frente a la aplicación de entrega de producto de repartidor). La independencia de aplicación es la disparidad principal entre una ontología y un esquema de datos clásico (por ejemplo, EER, ORM, UML) aunque cada uno captura conocimientos a nivel conceptual. Por ejemplo, muchos investigadores han confundido las ontologías con esquemas de datos, bases de conocimiento, o incluso programas lógicos. A diferencia de un esquema de datos conceptual o una base de conocimientos “clásica” que captura la semántica para una aplicación de empresa determinada, la ventaja principal y fundamental de una ontología es que captura el conocimiento del dominio de manera muy independiente de cualquier aplicación o tarea específica. Un consenso sobre el contenido ontológico es el principal requisito en la ingeniería de ontología, y esto es lo que la distingue principalmente del modelado de datos conceptual.

El área de investigación de la ingeniería de ontología parece haber alcanzado un cierto nivel de madurez, teniendo en cuenta la gran cantidad de métodos y herramientas contemporáneos para la formalización y la aplicación de modelos de representación del conocimiento. Sin embargo, todavía hay poca comprensión de y apoyo para los aspectos evolutivos de las ontologías. Esto es específicamente importante en los entornos distribuidos y colaborativos en los que las ontologías co-evolucionan de manera natural con sus comunidades de uso. Para gestionar la evolución de las ontologías individuales, se han adoptado con éxito unas técnicas establecidas a partir de la evolución del esquema de datos, y parece emerger un consenso sobre un modelo de proceso de evolución de ontología general.

Sin embargo, mucho menos explorado, es el problema de la evolución de las ontologías interorganizacionales. En este entorno dinámico y “complejo”, un modelo de proceso de cambio colaborativo requiere unas metodologías más poderosas de ingeniería, argumentación y negociación, complementadas por el apoyo de la gestión de dependencia de contexto. Resulta que se puede aprender mucho de otros dominios en los que se están diseñando objetos formales de manera colaborativa. En particular, el campo de la ingeniería de sistemas ofrece una gran cantidad de técnicas y herramientas para versionar, mezclar y evolucionar los objetos de software, y muchas de estas técnicas pueden reutilizarse en un entorno de ingeniería de ontología. El desafío pendiente clave es construir un marco único, basado en estos mecanismos, que pueda adaptarse a las necesidades de un entorno de versión específico.

Las ontologías a menudo reutilizan y amplían (partes de) otras ontologías. Existen muchos tipos diferentes de dependencias dentro y entre los objetos ontológicos de diversos niveles de granularidad, que van desde los conceptos individuales de las definiciones a ontologías completas. Otros objetos dependientes incluyen casos, así como los programas de aplicación que comprometen a la ontología. Por lo tanto, un cambio en la solicitud en un objeto podría implicar una cascada de cambios en respuesta a todos sus objetos dependientes. Se necesita un método viable para registrar todas las implicaciones de la ontología y sus objetos dependientes.

El documento EP1970844 describe un método para el metamodelado que usa unos objetos de ontología dinámica. Los llamados “objetos ontológicos”, es decir, los conjuntos de interfaces e implementaciones de software, se actualizan sobre la base de los cambios en un metamodelo: cuando se ha realizado un cambio en el metamodelo, los objetos ontológicos se actualizan generando un nuevo código fuente. Un problema en este caso es que el cambio se realiza en el nivel de los objetos ontológicos. No existe un mapeado entre el modelo de ontología y el metamodelo. Sigue siendo difícil para una aplicación participante detectar y aplicar los cambios correspondientes a

la interfaz entre la aplicación y los objetos ontológicos.

El mismo problema se produce en el documento US2007/162409, en el que se recibe una orden de un solicitante para modificar una tabla que incluye los conceptos de dominio y los primitivos semánticos relacionados con un dominio. La tabla se actualiza en respuesta a la orden. A continuación, la tabla actualizada se almacena como una ontología.

En el documento "Dogma-Mess: A Meaning Evolution Support System for Interorganizational Ontology Engineering" (A. De Moor et al., Conceptual Structures: Inspiration and Application Lecture Notes in Computer Science, vol. 4068, 1 de enero 2006, páginas 189-202) se introduce un modelo para la ingeniería de ontología interorganizacional. El documento se limita a describir la evolución de un sistema de ontología basado en un conjunto de especializaciones en esta ontología, estas últimas llamadas ontologías organizacionales. No describe los operadores de evolución con gran detalle. Más importante aún, no tiene en cuenta las aplicaciones mapeadas en el sistema de ontología. En otras palabras, no describe cómo evolucionan los mapeos entre rutas de aplicación en un sistema de datos arbitrario y las rutas conceptuales en un sistema de ontología basado en los cambios aplicados al sistema de ontología. El documento menciona las operaciones de versionado y de cambio, pero no describe esto de una manera formal y detallada.

El documento "Data Modelling versus Ontology engineering" (P. Spyns et al., ACM Sigmod Record, vol. 31, Nº 4, páginas 12-17, 31 de diciembre de 2002) describe la diferencia entre un modelo de datos y una ontología como se percibe por los autores, e introduce el enfoque de ontología DOGMA. No describe la evolución de las ontologías, ni describe cómo evolucionan los mapeos entre las rutas de aplicación en un sistema de datos arbitrario para las rutas conceptuales en un sistema de ontología basado en los cambios aplicados a ese sistema de ontología.

El documento "Object Role Modelling for Ontology Engineering in the DOGMA Framework" (P.Spyns et al., On the Move to Meaningful Internet Systems 2005, OTM Workshop Lecture Notes, vol. 3762, páginas 710 a 719, 1 de enero 2005) se limita a una descripción de algunas cuestiones básicas a tener en cuenta cuando se usa la metodología ORM para la ingeniería de ontología desde el punto de vista del marco de la ontología DOGMA. En primer lugar se trata la diferencia entre un modelo de datos y una ontología. Además se tratan las semánticas natural y formal. Por último, el documento trata algunas etapas básicas a tener en cuenta en la construcción de las ontologías. El documento no hace referencia a los temas de la evolución o el versionado.

En el documento US2008/071731 se presenta una solución para refinar de manera automática la ontología dentro de un contexto específico. En el método propuesto se usa un denominado extractor de contexto rico para descubrir un conflicto de relación semántica entre un esquema de ontología dado y unos datos de aplicación. Este extractor descubre que ciertos datos de aplicación contienen un contexto más rico que el esquema de ontología existente al que están mapeados. A continuación, el sistema de ontología se refina con el contexto más rico de los datos de aplicación y se define una nueva gráfica de mapeo entre la ontología refinada y los esquemas de datos originales.

Existe una necesidad de mantener y aplicar cambios a un sistema de ontología en el nivel de la interfaz entre el sistema de aplicación y de ontología. Esto es especialmente necesario cuando se enlazan múltiples aplicaciones al mismo sistema de ontología. La adaptación de los cambios puede ser, de otro modo, un trabajo muy elaborado.

Objetivos de la invención

La presente invención tiene como objetivo proporcionar un método y un sistema para la evolución de ontología flexible, con los que se puedan realizar un seguimiento de los cambios dentro del sistema de ontología.

Sumario de la invención

La presente invención propone actualizar los mapeos de las rutas de aplicación en un sistema de datos a las rutas conceptuales en un sistema de ontología basado en los cambios detectados en ese sistema de ontología. Hay que tener en cuenta que, a diferencia de los objetos de ontología en las soluciones de la técnica anterior (por ejemplo, el documento EP1970844), el sistema de datos en esta invención no tiene que cambiarse durante el proceso, se cambian solo los mapeos. Un sistema de ontología se considera que está enlazado con un sistema de datos usado por una aplicación informática. La ontología tiene una cierta estructura sintáctica, en la que se definen unas rutas conceptuales. El sistema de datos tiene una estructura direccionable por las rutas de aplicación. Se considera una situación en la que se ha hecho un cambio en el sistema de ontología.

Más específicamente, la presente invención proporciona un método para modificar un mapeo desde al menos una ruta de aplicación de un sistema de datos a una ruta conceptual de un sistema de ontología, direccionando dicha ruta de aplicación una parte de la estructura del sistema de datos y direccionando dicha ruta conceptual una parte de la estructura del sistema de ontología. El método comprende las etapas de:

- detectar un cambio en una parte de la estructura del sistema de ontología en el que se direcciona una o más de las rutas conceptuales;

- actualizar los mapeos para reflejar el cambio en esa parte de la estructura del sistema de ontología.

Algunos ejemplos de los cambios que se producen a menudo en una ontología (como, por ejemplo, reetiquetar, añadir, insertar, borrar) se proporcionan a continuación.

En una realización preferida, el cambio detectado se almacena en un registro de cambios. Un registro de cambios de este tipo realiza un seguimiento de los diversos cambios en el sistema de ontología al pasar de una versión a una versión sucesora.

En una realización, el método comprende una etapa inicial de aplicación de dicho cambio al sistema de ontología. Esta es una situación típica: primero se realiza un cambio en una parte de la estructura del sistema de ontología en cuestión. A continuación, dicho cambio debe detectarse y los mapeos necesitan una actualización. En una realización se imponen una o más condiciones para aplicar dicho cambio.

Un cambio en (una parte de) la estructura del sistema de ontología puede constituir un único cambio o un conjunto de varios cambios. En una realización el cambio comprende un reetiquetado de una parte de la estructura del sistema de ontología. El cambio también puede ser añadir otra estructura a parte de la estructura del sistema de ontología, no siendo dicha otra estructura parte de la estructura del sistema de ontología antes del cambio. El cambio comprende insertar una estructura adicional en una parte de la estructura del sistema de ontología, con lo que esa estructura adicional no es parte de la estructura del sistema de ontología antes del cambio. El cambio también puede comprender el borrado de parte de la estructura del sistema de ontología.

En otro aspecto, se proporciona un programa ejecutable en un dispositivo programable que contiene instrucciones, que, al ejecutarse, realizan el método que figura anteriormente.

En un aspecto adicional se proporciona un dispositivo para modificar un mapeo de al menos una ruta de aplicación de un sistema de datos a una ruta conceptual de un sistema de ontología, por lo que la ruta de aplicación direcciona una parte de la estructura de dicho sistema de datos y la ruta conceptual direcciona una parte de la estructura del sistema de ontología. El dispositivo comprende unos medios para detectar un cambio en una parte de la estructura del sistema de ontología que está direccionando una o más de dichas rutas conceptuales y medios para actualizar los mapeos para reflejar el cambio en la parte de la estructura del sistema de ontología. También pueden producirse las combinaciones de estos posibles cambios, como apreciarán los expertos en la materia.

En una realización preferida, el dispositivo está dispuesto para identificar y almacenar la información sobre la versión del sistema de ontología.

Breve descripción de los dibujos

La figura 1 ilustra en la parte superior un patrón semántico que podría ser útil para anotar el esquema de datos lógico en la parte inferior de la figura
La figura 2 representa una vista general de las entradas y las salidas del motor de traducción.
La figura 3 ilustra un tipo de hecho que puede desplazarse en dos direcciones, indicando dos rutas.
La figura 4 ilustra un cambio que comprende un reetiquetado, un añadido, una inserción y un borrado, respectivamente.
La figura 5 ilustra un patrón de semántica simple para expresar que las personas pueden tener nombres.
La figura 6 ilustra el patrón semántico de la figura 5 después de aplicar un registro de cambios para refinar el concepto nombre.
La figura 7 ilustra un patrón semántico para expresar que las personas pueden tener nombres y apellidos.
La figura 8 ilustra el patrón semántico de la figura 7 después de aplicar un registro de cambios para distribuir la semántica sobre tanto los términos como los roles.
La figura 9 ilustra el patrón semántico de la figura 6 después de aplicar un registro de cambios para distribuir la semántica sobre tanto los términos como los roles.
La figura 10 ilustra un patrón semántico para expresar que las personas son parte de un determinado grupo, y este grupo se localiza en una dirección determinada.
La figura 11 ilustra el patrón semántico de la figura 10 después de aplicar un registro de cambios para mover las relaciones.
La figura 12 ilustra un patrón semántico para expresar que un grupo tiene una línea de dirección y un país.
La figura 13 ilustra el patrón semántico de la figura 12 después de aplicar un registro de cambios para añadir una dirección de concepto que consiste en una línea de dirección y un país.
La figura 14 ilustra un patrón semántico para expresar que una persona tiene una dirección que tiene una línea de dirección.
La figura 15 ilustra el patrón semántico de la figura 14 después de aplicar un registro de cambios para cortocircuitar el concepto de direcciones.
La figura 16 ilustra un patrón semántico después de aplicar un registro de cambios para el rol/co-rol de una persona que tenga una dirección en una persona localizada en una dirección.

La figura 17 ilustra el patrón semántico de la figura 12 después de aplicar un registro de cambios para renombrar el término grupo por el término persona.

Descripción detallada de la invención

Las ontologías definen, en general, representaciones compartidas por dos aspectos esenciales y duales de la semántica de un dominio, es decir, una semántica formal de información que permita el procesamiento de la información por un ordenador y una semántica del mundo real que permita enlazar el contenido procesable por máquina con un significado para los seres humanos sobre la base de las terminologías consensuales y del lenguaje natural.

El enfoque de representación de ontologías introducido en la presente invención adopta un enfoque de modelado orientado a los hechos. En este enfoque, los hechos se consideran la unidad de comunicación y, por lo tanto, construyen representaciones de la semántica a partir de la abstracción de estos hechos observados en los tipos de hecho. Los hechos representan objetos (entidades o valores) que juegan un cierto rol (parte en la relación). Esto es diferente de los enfoques basados en los atributos tal como el modelado orientado a objetos, en el que el dominio está representado por la abstracción de los tipos de datos de los objetos observados.

Un sistema de datos se define por la combinación de una parte intensional y extensional. Su definición intensional prescribe las limitaciones de los elementos de datos en forma de tipos de datos (por ejemplo, números enteros, cadenas) y las relaciones entre los mismos (por ejemplo, la multiplicidad). Una definición extensional prescribe que los elementos de datos pertenecen a qué tipos de datos. Un sistema de datos puede anotarse por una ontología y puede considerarse como una aplicación que compromete a la ontología. Una aplicación compromete a un sistema de ontología si el sistema de datos está usando los mapas en el sistema de ontología. Un mapeo de este tipo se realiza por rutas de aplicación de mapeo que direccionan una parte de la estructura del sistema de datos. Un ejemplo de un sistema de datos es una base de datos relacional. En una base de datos relacional un ejemplo de una ruta de aplicación podría direccionar un atributo específico en una tabla específica. La parte intensional se define mediante su esquema. La parte extensional se define mediante su población. Otros ejemplos de sistemas de datos incluyen XML y su esquema correspondiente, los mensajes EDI, etc. El sistema de ontología también puede manejar los sistemas de datos para los que la parte intencional no está disponible, por ejemplo, XML que no tiene un esquema correspondiente. Por ejemplo, en un archivo XML una ruta de aplicación podría ser una XPath.

El sistema de ontología separa la representación de la semántica del dominio y la anotación de la aplicación con esta semántica en tres capas separadas: base de lexon, base de patrón, capa de compromiso.

Un lexon es un quíntuple que define un tipo de hecho binario plausible dentro de un cierto contexto. El quíntuple se define como {Contexto, término de cabecera, rol, co-rol, término de cola}. El rol, y su inverso el co-rol, define la relación conceptual entre el término de cabecera y de cola. Intuitivamente un lexon puede leerse como: dentro del contexto, el término de cabecera puede tener una relación con el término de cola en el que juega un rol, y a la inversa, en el que el término de cola juega un co-rol correspondiente.

Un contexto en un lexon proporciona una referencia para uno o más recursos léxicos y/o partes de un recurso léxico (tal como los documentos) a partir del que se ha obtenido el lexon. Los contextos son importantes para evitar las ambigüedades léxicas de los términos dentro de los lexones, ya que el significado de los términos puede variar de acuerdo con el contexto.

Los lexones como tales no tienen una dirección de lectura, sin embargo, cuando se construye una ruta se les pueden proporcionar una de dos direcciones, ya sea empezando por el término de cabecera o el término de cola.

Una base de lexon se define como un conjunto de lexones. El objetivo de la base de lexon es proporcionar un recurso compartido y en evolución que se use para llegar a un entendimiento común y acordado sobre el vocabulario de ontología y por lo tanto se dirige al entendimiento humano, asociando términos del lenguaje natural. La base de lexon refleja la estructura sintáctica del sistema de ontología.

Una ruta conceptual en una base de lexon se define por un par de términos de contexto o una concatenación finita de lexones en esa base de lexon. En este último caso también se impone una dirección de lectura específica. Una ruta conceptual puede ser un solo concepto (por ejemplo, la entrega), un lexon (por ejemplo, la fecha de entrega) o una agrupación de diferentes lexones (por ejemplo, el día de la fecha de la fecha y la hora de entrega del envío). Por lo tanto, las rutas conceptuales son capaces de direccionar las partes de la estructura del sistema de ontología. Las rutas conceptuales se usan en los patrones para definir las restricciones semánticas y en los compromisos para definir los mapeos.

Un patrón semántico se define por una selección significativa de las rutas conceptuales en una base de lexon específica más un conjunto de restricciones semánticas. Cada restricción semántica tiene cierto tipo y se expresa como una recopilación de conjuntos de rutas conceptuales. La figura 1 ilustra un patrón semántico en la notación ORM. Incluye diferentes rutas conceptuales. Por ejemplo, desde la parte inferior derecha del término Apellido hasta

la parte superior izquierda del término País, se verbaliza la ruta conceptual: Apellido parte del Nombre que identifica la Persona que reside en el País. Una base de patrón se define por un conjunto de patrones semánticos.

Un compromiso de una aplicación informática para un sistema de ontología se define por una selección de partes de patrones semánticos de la base de lexon en el sistema de ontología, un conjunto de restricciones semánticas sobre estos patrones y un conjunto de mapeos que se mapean desde el sistema de datos usado por la aplicación al sistema de ontología. Más precisamente, las rutas de aplicación del sistema de datos se mapean a las rutas conceptuales en estos patrones. La capa de compromiso se define por el conjunto de todos los compromisos.

Cada compromiso individual dentro de la capa de compromiso es una representación de la semántica de un sistema de datos específico en términos de la base de lexon. Los patrones pueden reutilizarse a través de compromisos si se seleccionan de la misma base de patrón. Si se hace, el sistema de ontología establece la interoperabilidad semántica entre los sistemas de datos y las aplicaciones en general (por ejemplo, los agentes de software y los servicios web).

Las reglas restringen y atribuyen interpretaciones específicas a un subconjunto seleccionado de patrones contenidos dentro de la base de patrón, por ejemplo, cada Persona tiene al menos una dirección. Como tal, cada regla de compromiso individual representa la semántica de una aplicación específica.

Las rutas de aplicación que direccionan la parte de la estructura del sistema de datos pueden mapearse en rutas conceptuales. Se requieren mapeos para crear de manera automática las transformaciones de valor de datos entre un sistema de datos estructurado y un sistema de ontología estructurado. Dado un sistema de datos de origen y un sistema de datos de destino, y sus respectivos mapeos para una parte compartida de un sistema de ontología, es posible crear de manera automática unas transformaciones de valor de datos entre dichos sistemas de datos de origen y de destino. La transformación se realiza de manera automática mediante un motor de traducción. El motor de traducción funciona analizando un compromiso de origen y de destino, que contiene los mapeos de las rutas de aplicación de un sistema de datos en las rutas conceptuales de un sistema de ontología, y lee los símbolos de datos del sistema de datos de origen y escribe los símbolos traducidos en el sistema de datos de destino. La figura 2 proporciona una visión general de las entradas y salidas del motor de traducción. El motor en sí mismo está compuesto de:

- un analizador que construye un árbol sintáctico a partir de un compromiso textual;
- un lector que se encarga de consultar un sistema de datos;
- un escritor que se encarga de la actualización y la creación de datos en un sistema de datos;
- un motor de secuenciación de órdenes accesible por el lector y el escritor.

Las secuenciaciones de órdenes personalizadas pueden escribirse por los usuarios del motor que permiten la modificación del comportamiento en tiempo de ejecución sin cambiar el propio motor. El motor de traducción usa un enfoque de impulso y de arrastre en función del tipo de sistema de datos.

Enfoque de impulso

En el enfoque de impulso el Lector comienza a leer el sistema de datos de origen e impulsa los símbolos de datos hacia el sistema de almacenamiento, que se aceptan inmediatamente por el Escritor y se escriben en el sistema de datos de destino. En efecto, el sistema de datos de origen está impulsando sus datos en el sistema de datos de destino. El enfoque de impulso se usa para los sistemas de datos de origen con estructura de árbol, tal como XML. En el compromiso de origen los mapeos se procesan en el orden en que se listan. El primer mapeo se asume para que apunte al elemento raíz.
Los mapeos deben estar en la siguiente forma:

map “/a” on A.

map “/a/b” on B de A.

map “/a/b/c” on C de B de A.

map “/a/d” on D de A.

La palabra clave ’map’ indica que esta declaración es un mapeo. La parte izquierda del mapeo es una ruta de aplicación expresada en un lenguaje de ruta que puede usarse para direccionar los elementos en el sistema de datos (por ejemplo, XPath). La ruta de aplicación está encerrada entre comillas dobles.

La ruta de aplicación contiene suficiente información para construir una consulta procesable por un motor de consulta que soporte el formato de datos del sistema de datos. El Lector envía estas consultas al motor de consulta adecuado. Por ejemplo, en XML, las XPaths pueden procesarse como consultas por un motor Xpath como Xalan. La misma ruta de aplicación se usa también por el Escritor para escribir los datos en la localización a la que se dirige la

ruta.

La palabra clave 'on' separa las partes izquierda y derecha de la declaración de mapeo. A la derecha está escrita una ruta conceptual, construida usando los símbolos del sistema de ontología. Cada declaración termina con un punto.

Los mapeos se procesan en el orden dado. Las rutas conceptuales están relacionadas entre sí por este orden. En el ejemplo anterior la instancia para el concepto *A* es la misma en cada uno de los mapeos. Esto se conoce porque el primer elemento en la ruta de aplicación es también el mismo y porque está apuntando al elemento raíz. Otros elementos no raíz pueden repetirse y cada vez se instancia el concepto correspondiente. Por ejemplo, cuando hay tres elementos *b*, se instancian tres conceptos *B* correspondientes. Para cada uno de los mapeos pueden instanciarse cero o más rutas conceptuales, en función del número de elementos en el sistema de datos de origen.
A continuación está un ejemplo de XML que es válido para los mapeos anteriores.

```
<a>
    <b>
        <c> texto </c>
    </b>
    <b>
        <c> texto2 </c>
    </b>
    <b>
        <c> texto3 </c>
    </b>
</a>
```

Esto instancia las siguientes rutas (las instancias de los conceptos están entre paréntesis, donde las entidades se identifican por un número prefijado con el símbolo @, y los valores se muestran entre comillas dobles):

A (@ 1).
B (@ 1) of A (@ 1).
B (@ 2) of A (@ 1).
B (@ 3) of A (@ 1).
C ("texto") of B (@ 1) of A (@ 1).
C ("texto2") of B (@ 2) of A (@ 1).
C ("texto3") of B (@ 3) of A (@ 1).

El escritor usa el reverso de este proceso y construye los datos para el sistema de datos de destino a partir de estas rutas instanciadas. El escritor consulta qué mapeo corresponde a la ruta instanciada y construye los elementos en el sistema de datos.

Enfoque de arrastre

Como alternativa, un enfoque de arrastre permite al Escritor decidir qué datos necesita del Lector. En este caso, el Escritor pregunta al Lector por cada pieza de información que necesita. El Escritor está arrastrando los datos del lector, que los coloca en el almacenamiento, listos para que acceda el Escritor. El enfoque de arrastre se usa para los sistemas de datos basados en gráficas, tales como las bases de datos relacionales. En contraposición con el enfoque de impulso, el enfoque de arrastre comienza con el escritor que solicita las rutas instanciadas del lector por sus mapeos.

Dadas unas consultas conceptuales de compromiso ontológico se ejecutan en cualquier sistema de datos que está anotado correctamente por ese compromiso. Una consulta conceptual es una construcción de lenguaje expresada en términos de una o más rutas conceptuales en un compromiso. Dados los mapeos entre las rutas conceptuales y las rutas de aplicaciones en un compromiso, las consultas conceptuales pueden ejecutarse como consultas lógicas en el sistema de datos de compromiso. Esto se hace traduciendo (las rutas conceptuales en) la consulta conceptual en una consulta lógica en términos de las rutas de aplicación del sistema de datos. Una consulta lógica es una consulta dentro del sistema de datos. Por ejemplo, una consulta lógica en una base de datos relacional se expresaría en SQL.

En una realización se especifican los compromisos ontológicos en el lenguaje natural controlado Ω-RIDL (Omega-RIDL). El lenguaje describe reglas semánticas en términos de rutas conceptuales en el que las etiquetas de rol y co-rol necesitan interpretarse como una relación ontológica. Ω-RIDL es tanto una consulta conceptual como un lenguaje de especificación de compromiso.

Los sistemas y los métodos de la invención pueden verse como una mejora adicional del enfoque DOGMA para el desarrollo de la mediación de ontología guiada de los agentes. DOGMA (R. Meersman., Proc. of the International

Symposium on Methodologies for Intelligent Systems (ISM IS), páginas 30-45, 1999) es un enfoque ontológico y un marco que no se restringe a un lenguaje de representación específico. Las realizaciones de la presente invención aplican el marco DOGMA a la ontología sobre la base del mapeo de datos.

La figura 3 ilustra una relación entre dos conceptos Producto y Orden. Cada lexon puede recorrerse en dos direcciones, denotando dos rutas: un Orden para un producto, y un producto que tiene un Orden.

Las rutas conceptuales pueden concatenarse para formar rutas compuestas. Estas concatenaciones pueden formalizarse de manera adicional con reglas y limitaciones, que restringen el posible uso de los conceptos y las relaciones en la ontología.

La anotación se expresa en Ω-RIDL. La figura 1 ilustra, en la parte superior, un patrón semántico que podría ser usado para anotar el esquema lógico de datos en la parte inferior de la figura.

Cada símbolo relevante en la instancia de esquema de datos lógicos se anota por una ruta conceptual significativa en el patrón. Si un símbolo relevante no puede anotarse por la versión de patrón actual, el patrón se cambia de tal manera que la nueva versión de patrón permite anotar los símbolos pendientes.

En el ejemplo anterior, todos los atributos de la tabla Personas (las rutas de aplicación) pueden mapearse por rutas conceptuales en el patrón. Por lo tanto, se puede leer:

- map “Gente.nombre” on “Nombre parte del Nombre que identifica a la Persona”
- map “Gente.apellido” on “Apellido parte del Nombre que identifica a la Persona”
- map “Gente.ciudad” on “Ciudad es el lugar de nacimiento de la Persona”
- map “Gente.país” on “País donde reside la Persona”

El ejemplo anterior es trivial, ya que los términos de los símbolos de atributos son interpretables de manera intuitiva. Sin embargo, en escenarios del mundo real el significado de los símbolos en el esquema lógico está, en general, implícito. Incluso en este ejemplo: aunque los significados independientes de los símbolos ciudad, país y persona son intuitivamente obvios, su interrelación no lo es. País y ciudad parecen no estar relacionados en absoluto, ya que también se puede deducir a partir de los valores de los atributos. Por otra parte, nombre y apellido parecen estar relacionados entre sí de manera indirecta a través de la relación (parte de) metonímica con nombre.

El sistema de anotación anterior funciona para diferentes tipos de tecnologías de gestión de datos, incluyendo las tablas y las columnas relacionales, las clases orientadas a objetos, o las etiquetas XML. Dada una ontología compartida, las consultas en un esquema de datos lógico con anotaciones pueden traducirse de manera automática a consultas en cualquier otro esquema de datos lógico que se anota con esta ontología. Por lo tanto, esto proporciona un enfoque universal para la integración de datos. Además, como las anotaciones están cerca del lenguaje natural, el significado de los símbolos en los esquemas puede describirse e interpretarse por el usuario final lego en esta materia.

Evolución

La implementación de un cambio en un sistema de ontología es un proceso difícil que necesita muchas diferentes sub-actividades: propagación del cambio, reestructuración y gestión de la inconsistencia.

Con este fin, el dispositivo de acuerdo con la presente invención comprende en una realización preferida una capa de versionado. Dicha capa de versionado proporciona toda la funcionalidad para identificar y almacenar las versiones de ontología y los registros de cambios. Los registros de cambios realizan un seguimiento de los cambios necesarios para determinar una versión del sistema de ontología empezando a partir de una versión anterior. Un editor de ontologías puede soportar la edición versionada de las ontologías con una interfaz gráfica.

Toda la información sobre los cambios realizados se sigue y se registra en un registro de cambios, también denominado como un registro de versión. Un registro de cambios entre una versión K del sistema de ontología y una versión K + 1 del sistema de ontología se define como una secuencia de operaciones de cambio que se realizan en la versión K, con el fin de resultar en la versión K + 1.

Los ejemplos de los cambios a nivel de base de lexon se reetiquetan, insertando o añadiendo lexones a o eliminando lexones de la estructura del sistema de ontología existente. Ejemplos de operadores de cambio en el nivel de base de patrón son crear o eliminar un patrón. Ejemplos de operadores de cambio en el nivel de la capa de compromiso son crear, revisar o eliminar una ruta o una regla. Esto se ilustra en la figura 4. La figura muestra de arriba hacia abajo cuatro tipos de cambios que pueden ocurrir en un sistema de ontología. El tipo reetiquetar incluye cualquier renombrado de un término para un concepto (nodo) o relación (borde) en el sistema de ontología. En el primer ejemplo, antes del cambio un concepto se denomina por un término ‘*B’*. Después del cambio el mismo concepto se denomina por un término ‘*B’’*. En el segundo ejemplo, la relación denominada por ‘a’ entre los conceptos (denominada por los términos ‘*A’* y ‘*B’*) consigue un nuevo término ‘*a’’*.

El tipo añadir incluye cualquier cambio que se asocia a una nueva estructura del sistema de ontología existente. Por ejemplo, antes del cambio, una parte del sistema de ontología consiste en un concepto denominado por el término 'A'. Después del cambio se añade una nueva estructura que incluye los conceptos denominados por los términos '*B'* y '*C'* y las relaciones denominadas por 'a' y 'b', respectivamente, a este concepto ya existente.

El tipo insertar incluye cualquier cambio que se inserta en una nueva estructura entre dos partes existentes del sistema de ontología. Por ejemplo, en este caso se inserta un nuevo concepto denominado por el término '*B'* y la relación denominada por '*c'* entre los conceptos existentes '*A'* y '*C'*. El tipo insertar se aplica en los ejemplos tratados a continuación Mover Relación, Conceptualización, Cortocircuito.

El tipo borrar incluye cualquier cambio que elimina una parte de la estructura del sistema de ontología. Por ejemplo, la eliminación de una relación que se denomina por el término *'a'* entre los conceptos '*A'* y '*B'*.

Actualizar mapeos basándose en cambiar el sistema de ontología

Un mapeo de un sistema de datos a una versión K del sistema de ontología podría romperse cuando se cambia la versión K del sistema de ontología a la versión K + 1, porque las rutas conceptuales usadas en el mapeo se reestructuran, se borran o se desaprueban. Por lo tanto cualquier mapeo con la versión K del sistema de ontología debería adaptarse a la nueva versión K + 1 del sistema de ontología. El registro de cambios almacena la secuencia de todas las operaciones (cambios) entre dos versiones de sistema de ontología. El registro de cambios puede usarse de este modo para diferentes fines tales como para identificar conflictos o divergencias de significado. En el ejemplo anterior, el cambio de símbolo "Persona" a "Individual" desencadena un cambio en el mapeo en el que ha estado involucrado Persona:

- map "Gente.nombre" on "Nombre parte del Nombre que identifica al Individuo"
- map "Gente.apellido" on "Apellido parte del Nombre que identifica al Individuo"
- map "Gente.ciudad" on "Ciudad es el lugar de nacimiento del Individuo"
- map "Gente.país" on "País donde reside del Individuo"

Usando el registro de cambios de otros conflictos puede identificarse cuándo mezclar las evoluciones paralelas de una versión ontología. Para cada uno de los conflictos, la presente invención proporciona un conjunto de estrategias de resolución alternativas.

Derivar un conjunto de patrones reutilizables en la Ontología

Dado un conjunto de ontologías similares, las partes comunes pueden recopilarse y abstraerse de manera automática en patrones que pueden reutilizarse para fines posteriores. Por ejemplo, diferentes organizaciones pueden reutilizar de manera independiente el patrón de concepto de una tarea de trabajo para anotar sus esquemas de datos acerca de la entrega.

Un registro de cambios es una trascripción de un proceso de evolución de ontología que está representada por una secuencia ordenada de operaciones de ontología. Cada registro de cambios se limita a las operaciones de algunas clases de operadores específicos. Algunos operadores tienen pre y post-condiciones y cada operador pertenece a una clase de operador. También se registran las operaciones fallidas, incluyendo las condiciones insatisfechas que provocaron el fallo.

Un operador de evolución de ontología atómico C define la semántica de cambio de una ontología en términos de las siguientes propiedades: una condición previa PreCond (C) para un operador de cambio C es una condición que debe mantenerse en una ontología O con el fin de que C pueda aplicarse a esa ontología O, y una postcondición PostCond (C) para un operador de cambio C es una condición que debe mantenerse en una ontología O' que es el resultado de aplicar C a la ontología O.

Un operador de evolución de ontología compuesto se define por una secuencia ordenada de operadores de evolución de ontología atómicos. Una operación de evolución de ontología es una ocurrencia específica de un operador de evolución de ontología.

En la presente invención, los cambios proporcionados se detectan para la anotación de uno o más sistemas de información, comprendiendo dichos cambios un conjunto de operadores compuestos que agrupan un número de operadores atómicos de los patrones semánticos y los operadores equivalentes para los mapeos en los compromisos, en los que dichos operadores compuestos se añaden al registro de cambios.

A continuación, se explican las realizaciones específicas de las operaciones y los operadores del cambio por medio de ejemplos. Otros conjuntos de operadores pueden recibirse por el registro de cambios para proporcionar una secuencia de cambios. Cada uno de los operadores es una variación de los tipos de cambios mencionados anteriormente: reetiquetar, insertar, añadir y eliminar.

Para mayor comodidad del lector, el término de cabecera y de cola se inician con una letra mayúscula, y los términos rol/co-rol están en letras minúsculas.

Ejemplo 1: Refinar conceptos

En una realización, dichos operadores compuestos comprenden unos operadores atómicos para refinar los conceptos de refinación y actuar sobre la parte de la ruta conceptual de los mapeos correspondientes. El refinado es un ejemplo del tipo añadir.

Considérese el ejemplo ilustrado en las figuras 5 y 6. La figura 5 muestra un patrón semántico simple para expresar que las personas pueden tener nombres. Esto está mapeado para un ejemplo XML usando las siguientes expresiones Ω-RIDL:

```
map "concat (/persona/pname/fname, /persona/pname/lname)"
on Nombre de la Persona.
```

Debido a que el patrón semántico es demasiado limitado para expresar la información de, por ejemplo, un nuevo socio de negocios, debe ampliarse para incluir también el nombre y el apellido de manera explícita. Esto puede lograrse usando el siguiente registro de cambios:

```
introduceTerm ("Nombre")
defineDifferentia ("Nombre", "consiste en", "parte de", "Nombre")
introduceTerm ("Apellido")
defineDifferentia ("Nombre", "consiste en", "parte de", "Apellido")
introduceTerm (X)
defineDifferentia ("Nombre", r, r', X)
```

La aplicación de estas operaciones en el patrón semántico de la figura 5 lleva al patrón semántico de la figura 6. Este patrón semántico requiere un nuevo compromiso con los siguientes mapeos que reemplazan a los anteriores:

```
map "/persona/pname/fname"
on Nombre parte del Nombre de la Persona.
map "/persona/pname/lname"
on Apellido parte del Nombre de la Persona.
map "/persona/pname/xxx"
on X r' Nombre de la Persona.
```

La solución proporcionada por la presente invención es en primer lugar la introducción de un operador compuesto para el patrón semántico que puede incorporarse en el registro de cambios. El operador compuesto define una secuencia específica de los operadores atómicos. En segundo lugar, este operador compuesto necesita tener un operador equivalente que opere sobre los mapeos en el compromiso(s).

El nuevo operador puede definirse de la siguiente manera:

```
refine (X, r, r', Z1,... Zn)
  + introduceTerm (Z1)
  + defineDifferentia (X, r, r', Z1)
  + ...
  + introduceTerm (Zn)
  + defineDifferentia (X, r, r', Zn)
```

Aplicado en el ejemplo, esto proporciona el siguiente operador compuesto para el patrón semántico en el registro de cambios:

```
refine ("Nombre", "consiste en", "parte de", "Nombre", "Apellido", "X")
  + introduceTerm ("Nombre")
+ defineDifferentia ("Nombre", "consiste en", "parte de", "Nombre")
  + introduceTerm ("Apellido")
+ defineDifferentia ("Nombre", "consiste en", "parte de", "Apellido")
  + introduceTerm ("X")
+ defineDifferentia ("Nombre", "consiste en", "parte de", "X")
```

En cuanto a los mapeos, el operador actúa sobre la parte de la ruta conceptual de los mapeos:

```
map "concat (/persona/pname/fname, /persona/pname/lname)"
on Nombre de la Persona.
```

El operador expresa que esta ruta conceptual se ha refinado, y pregunta al usuario cómo realizar el refinamiento:

```
map "/persona/pname/fname"
on Z1... Zn ? r' Nombre de la Persona.
map "/persona/pname/lname"
on Z1... Zn ? r' Nombre de la Persona.
```

Obsérvese que el operador solo permite el refinamiento usando la misma relación, es decir, en el caso del ejemplo: consiste en/parte de. Una multitud de operadores "refine" pueden aplicarse, por ejemplo, "split" y "atribute".

Obsérvese también que la ruta de aplicación de XML en este ejemplo es una función agregada (concat). El algoritmo usado puede identificarles y por lo tanto, dividirlos perfectamente en mapeos separados (basándose en el operador "refine").

Ejemplo 2: Rolificar conceptos

En otra realización, los operadores compuestos comprenden unos operadores atómicos para distribuir la semántica sobre tanto los términos como los roles. Este es un ejemplo del tipo reetiquetar.

Considérese el ejemplo ilustrado en las figuras 7 y 8. La figura 7 describe el patrón semántico tal cual. El mapeo correspondiente es de la siguiente manera:

```
map "/persona/Nombre/fname"
on Nombre de la Persona.
map "/persona/Nombre/lname"
on Apellido de la Persona.
```

Debido a que se prefiere distribuir la semántica sobre tanto los términos como los roles, el patrón semántico necesita evolucionar usando el siguiente registro de cambios:

```
defineDifferentia ("Persona", "tiene nombre", "nombre de", "Nombre")
dropDifferentia ("Persona", "tiene", "de", "Nombre")
defineDifferentia ("Persona", "tiene apellido", "apellido de", "Nombre")
dropDifferentia ("Persona", "tiene", "de", "Apellido")
```

Esto resulta en el patrón de la figura 8, y debería resultar en mapeos actualizados en el compromiso de la siguiente manera:

```
map "/persona/Nombre/fname"
on Nombre nombre de la Persona.
map "/persona/Nombre/lname"
on Nombre apellido de la Persona.
```

La solución proporcionada por la presente invención es en primer lugar la introducción de un operador compuesto para el patrón semántico que puede incorporarse en el registro de cambios. El operador compuesto define una secuencia específica de los operadores atómicos. En segundo lugar, este operador compuesto necesita tener un operador equivalente que opere sobre los mapeos en el compromiso(s).
El nuevo operador se define de la siguiente manera:

```
rolify (X, r', r, Y, a', a)
  + defineDifferentia (X, a', a, Y)
  + dropDifferentia (Y, r', r, X)
o
rolify (X, r', r, Y)
  + rolify (X, r', r, Y, "X de", "tiene X")
```

Aplicado en el ejemplo, esto proporciona el siguiente elemento en el registro de cambios:

```
rolify ("Nombre", "de", "tiene", "persona", "nombre de", "tiene nombre")
  + defineDifferentia ("Nombre", "nombre de", "tiene nombre", "Persona")
  + dropDifferentia ("Nombre", "de", "tiene", "Persona")
o
rolify ("Nombre", "de", "tiene", "Persona")
  + defineDifferentia ("Nombre", "Nombre de", "tiene nombre", "Persona")
  + dropDifferentia ("Nombre", "de", "tiene", "Persona")
```

En cuanto a los mapeos, el operador actúa sobre la parte de la ruta conceptual de los mapeos:

```
map "/persona/pname/fname"
on Nombre de la Persona.
```

El operador expresa que esta ruta conceptual se ha rolificado, y realiza la operación

```
map /persona/Nombre/fname
on Nombre nombre de la Persona
```

Considérese otro ejemplo ilustrado en las figuras 6 y 9. La figura 6 describe el patrón semántico tal cual. El mapeo correspondiente es de la siguiente manera:

```
map "/persona/nombre/fname"
on Nombre parte del Nombre de la Persona.
map "/persona/nombre/lname"
on Apellido parte del Nombre de la Persona.
map "/persona/nombre/xxx"
on X r' Nombre de la persona.
```

Debido a que se prefiere para distribuir la semántica sobre tanto los términos como las funciones, el patrón semántico necesita evolucionar usando el siguiente registro de cambios:

```
defineDifferentia ("Persona", "tiene nombre", "nombre de", "Nombre")
  dropDifferentia ("Nombre", "consiste en", "parte de", "Nombre")
  defineDifferentia ("Persona", "tiene apellido", "apellido de", "Nombre")
  dropDifferentia ("Nombre", "consiste en", "parte de", "Apellido")
  defineDifferentia ("Persona", "tiene X", "X de", "Nombre")
   dropDifferentia ("Nombre", r, r', X)
   dropDifferentia ("Persona", "tiene", "de", "Nombre")
```

Esto resulta en el patrón de la figura 9, y debería resultar en unos mapeos actualizados en el compromiso de la siguiente manera:

```
map "/persona/nombre/fname"
on Nombre nombre de la Persona.
map "/persona/nombre/lname"
on Nombre apellido de la Persona.
map "/persona/nombre/xxx"
on Nombre X de la Persona.
```

La solución proporcionada por la presente invención es en primer lugar la introducción de un operador compuesto para el patrón semántico que puede incorporarse en el registro de cambios. El operador compuesto define una secuencia específica de los operadores atómicos. En segundo lugar, este operador compuesto necesita tener un operador equivalente que opere sobre los mapeos en el compromiso(s). El nuevo operador se define de la siguiente manera:

```
rolify (X, r', r, Y, a', a, Z)
  + definedifferentia (Z, "tiene X", "X de", Y)
    + dropdifferentia (Y, r', r, Z)
      [+ dropdifferentia (Z, a, a', Y)]
```

Aplicado en el ejemplo, se tiene el elemento siguiente en el registro de cambios:

```
+ rolify ("Nombre", "parte de", "parte de", "Nombre", "de", "tiene", "Persona")
   + definedifferentia ("Persona", "tiene nombre", "nombre de", "nombre de")
   + dropdifferentia ("Nombre", "consiste en", "parte de", "Nombre")
   [+ dropdifferentia ("Persona", "tiene", "de", "Nombre")]
+ rolify ("Apellido", "parte de", "parte de", "Nombre", "de", "tiene", "Persona")
   + definedifferentia ("Persona", "tiene el apellido", "apellido de", "nombre")
   + dropdifferentia ("Nombre", "consiste en", "parte de", "Apellido")
   [+ dropdifferentia ("Persona", "tiene", "de", "Nombre")]
+ rolify ("X", "parte de" "parte de", "Nombre", "de", "tiene", "Persona")
   + definedifferentia ("Persona", "tiene X", "X de", "Nombre")
   + dropdifferentia ("Nombre", "consiste en", "parte de", "X")
   [+ dropdifferentia ("Persona", "tiene", "de", "Nombre")]
```

En cuanto a los mapeos, el operador actúa sobre la parte de la ruta conceptual de los mapeos:

```
map "/persona/pname/fname"
on Nombre parte del Nombre de la Persona.
```

El operador expresa que esta ruta conceptual se ha rolificado, y realiza la operación

```
map "/persona/nombre/fname"
on Nombre nombre de la Persona.
```

Ejemplo 3: Mover relaciones

En una realización siguiente, dichos operadores compuestos comprenden unos operadores atómicos para mover las relaciones. Una relación de movimiento combina los tipos insertar y borrar descritos anteriormente. Considérese el ejemplo ilustrado en las figuras 10 y 11. El patrón semántico de la figura 10 explica que las personas son parte de un determinado grupo, y que este grupo se localiza en una dirección determinada. El patrón se ha comprometido con los siguientes mapeos:

```
map "/.../dirección"
on Dirección de Grupo.
```

Parecía que se había realizado un error en el diseño del patrón semántico, ya que la dirección debe ser en realidad la localización de la persona que es parte del grupo. Esto puede resolverse usando el siguiente registro de cambios:

```
defineDifferentia ("Persona", "localizado en", "localización de", "Dirección")
dropDifferentia ("Grupo", "localizado en", "localización de", "Dirección")
```

El resultado es el patrón semántico de la figura 11 y el compromiso correspondiente:

```
map "/.../dirección"
on Dirección de la Persona.
```

La solución proporcionada por la presente invención es en primer lugar la introducción de un operador compuesto para el patrón semántico que puede incorporarse en el registro de cambios. El operador compuesto define una secuencia específica de los operadores atómicos. En segundo lugar, este operador compuesto necesita tener un operador equivalente que opere sobre los mapeos en el compromiso(s).
El nuevo operador se define de la siguiente manera:

```
move (X, r', r, Y, Z)
   + defineDifferentia (X, r', r, Z)
   + dropDifferentia (X, r', r, Y)
```

que se ve de la siguiente manera para el ejemplo:

```
move ("Dirección", "localización de", "localizado de", "Grupo", "Persona")
  + defineDifferentia ("Dirección", "localización de", "localizado en", "Persona")
  + dropDifferentia ("Dirección", "localización de" "localizado en", "Grupo")
```

En cuanto a los mapeos, el operador actúa sobre la parte de la ruta conceptual de los mapeos:

```
map "/.../dirección"
on Dirección del Grupo.
```

El operador expresa que esta ruta conceptual se ha movido, y realiza la operación

```
map "/.../dirección"
on Dirección de la Persona.
```

Relacionado con esto está el nuevo operador de copia que se define de la siguiente manera:

```
copy (X, r', r, Y, Z)
  defineDifferentia (X, r', r, Y, Z)
```

Ejemplo 4: Conceptualización

En una realización adicional, dichos operadores compuestos comprenden unos operadores atómicos para añadir conceptos. Los tipos insertar y borrar se combinan de este modo.

Considérese el ejemplo ilustrado en las figuras 12 y 13. Considerando el patrón semántico de la figura 12 y el compromiso correspondiente:

```
map "/.../línea de dirección"
on Línea de Dirección del Grupo.
map "/.../país_nombre"
on País del Grupo.
```

Supóngase que el país y la línea de dirección necesitan estar relacionados a través de una nueva dirección de concepto. A continuación la dirección sirve como conceptualización intermedia de la relación entre país y la línea de dirección; y el grupo respectivamente.

Esto puede resolverse usando el siguiente registro de cambios:

```
introduceTerm ("Dirección")
defineDifferentia ("Grupo", "se localiza en", "localización de", "Dirección")
defineDifferentia ("Dirección", "consiste en", "parte de", "Línea de Dirección")
dropDifferentia ("Grupo", "tiene", "de", "Línea de Dirección")
defineDifferentia ("Dirección", "consiste en", "parte de", "País")
dropDifferentia ("Grupo", "tiene", "de" "Línea de Dirección")
```

El resultado es el patrón semántico de la figura 13 y el compromiso correspondiente:

```
map "/.../línea de dirección"
on la parte Línea Dirección de la localización de Dirección del Grupo.
map "/.../país_nombre"
on parte de País de la localización de Dirección del Grupo.
```

La solución proporcionada por la presente invención es en primer lugar la introducción de un operador compuesto para el patrón semántico que puede incorporarse en el registro de cambios. El operador compuesto define una secuencia específica de los operadores atómicos. En segundo lugar, este operador compuesto necesita tener un operador equivalente que opere sobre los mapeos en el compromiso(s).

El nuevo operador se define de la siguiente manera:

```
conceptify (X, r'0, r0, Y, r'n, rn, Zn)
  introduceTerm (X)
  defineDifferentia (X, r'0, r0, Y)
  move (Zn, r'n, rn, Y, X)
```

que se ve de la siguiente manera en el ejemplo:

```
conceptify ("Dirección", "localización de", "localizado en", "Grupo", "tiene", "de",
"Línea de dirección", "tiene", "de", "País")
    + introduceTerm ("Dirección")
   + defineDifferentia ("Dirección", "localización de", "localizado en", "Grupo")
  + move ("Línea de dirección", "de", "tiene", "Grupo", "Dirección")
    + move ("País", "de", "tiene", "Grupo", "Dirección")
```

En cuanto a los mapeos, el operador actúa sobre la parte de la ruta conceptual de los mapeos:

```
map "/.../línea de dirección"
on Línea de Dirección del Grupo.
```

El operador expresa que esta ruta conceptual se ha movido, y realiza la operación

```
map "/.../dirección"
on Línea de Dirección de la localización de la Dirección del Grupo.
```

Ejemplo 5: Cortocircuitar conceptos

En otra realización, dichos operadores compuestos comprenden operadores atómicos para eliminar conceptos. De este modo, se combinan el tipo insertar y eliminar como se ha descrito anteriormente.

Considérese el ejemplo ilustrado en las figuras 14 y 15. Considerando el patrón semántico de la figura 14 y el compromiso correspondiente:

```
map /persona/dirección/línea de dirección
on Línea de Dirección de localización de Dirección de la Persona
```

Supóngase que la ruta mapeada “Línea de Dirección de localización de Dirección de la Persona” necesita simplificarse a “Línea de Dirección de localización de la persona”.
Esto puede resolverse aplicando el siguiente registro de cambios:

```
defineDifferentia (“Persona”, “localizado en”, “localización de”, “Línea de Dirección”)
dropDifferentia (“Dirección”, “consiste en”, “parte de”, “Línea de Dirección”)
dropDifferentia (“Persona”, “localizado en”, “localización de”, “Dirección”)
```

El resultado es el patrón semántico de la figura 15 y el compromiso correspondiente:

```
map “/persona/dirección/línea de dirección”
on Línea de Dirección de localización de la persona
```

La solución proporcionada por la presente invención es en primer lugar la introducción de un operador compuesto para el patrón semántico que puede incorporarse en el registro de cambios. El operador compuesto define una secuencia específica de los operadores atómicos. En segundo lugar, este operador compuesto necesita tener un operador equivalente que opere sobre los mapeos en el compromiso(s).
El nuevo operador se define de la siguiente manera:

```
shortcircuit (X, r1, r’1, Y, r2, r’2, Z)
  + move (X, r1, r’1, Y, Z)
  + dropDifferentia (Y, r2, r’2, Z)
```

Que se ve de la siguiente manera en el ejemplo:

```
shortcircuit (“Persona”, “localizado en”, “localización de”, “Dirección”, “consiste en”, “parte de”, “Línea de Dirección”)
+ move (“Persona”, “localizado en”, “localización de, “Dirección”, “Línea de Dirección”)
+ dropDifferentia (“Dirección”, “consiste en”, “parte de”, “Línea de Dirección”)
```

En cuanto a los mapeos, el operador actúa sobre la parte de la ruta conceptual de los mapeos:

```
map “/.../línea de dirección”
on la Línea de dirección de localización de Dirección de la Persona.
```

El operador expresa que esta ruta conceptual se ha movido, y realiza la operación

```
map “/.../dirección”
on la localización la de Línea de dirección del Grupo.
```

Ejemplo 6: Renombrar el rol básico

En otro ejemplo más, dichos operadores compuestos comprenden unos operadores atómicos para renombrar los roles y/o los co-roles. Este es otro caso en el que se produce el tipo reetiquetar.

Considérese el ejemplo ilustrado en la figura 16. El siguiente patrón y el mapeado muestran la situación de inicio:

```
map “/persona/dirección”
on Dirección de la Persona.
```

Para hacer la semántica de este patrón más clara, se ejecuta el siguiente registro de cambios, por ejemplo, para hacer una diferencia con los diferentes tipos de localizaciones: “Persona nacida en/lugar de nacimiento de Dirección”, “Persona domiciliada en/domicilio de Dirección”,...

```
dropDifferentia (“Persona”, “tiene”, “de”, “Dirección”)
defineDifferentia (“Persona”, “localizado en”, “localización de”, “Dirección”)
```

El resultado de este registro de cambios se muestra en la figura 16, necesitando el siguiente mapeo correspondiente:

```
map “/persona/dirección”
on la localización de Dirección de la Persona.
```

La solución proporcionada por la presente invención es en primer lugar la introducción de un operador compuesto para el patrón semántico que puede incorporarse en el registro de cambios. El operador compuesto define una secuencia específica de los operadores atómicos. En segundo lugar, este operador compuesto necesita tener un operador equivalente que opere sobre los mapeos en el compromiso(s).
El nuevo operador se define de la siguiente manera:

```
renameDifferentia (X, r1, r’1, Y, r2, r’2)
  + dropDifferentia (X, r1, r’1, Y)
  + defineDifferentia (X, r2, r’2, Y)
```

que se ve de la siguiente manera para el ejemplo:

```
renameDifferentia (“Persona”, “tiene”, “de”, “Dirección”, “localizado en”,
“Localización de”)
  + dropDifferentia (“persona”, “tiene”, “de”, “Dirección”)
  + defineDifferentia (“Persona”, “localizado en”, “localización de”, “Dirección”)
```

En cuanto a los mapeos, el operador actúa sobre la parte de la ruta conceptual de los mapeos:

```
map “/persona/dirección”
on Dirección de la Persona.
```

El operador expresa que en esta ruta conceptual se ha renombrado, y realiza la operación

```
map “/persona/dirección”
on localización de Dirección de la Persona.
```

Ejemplo 7: Renombrar término básico

En una realización, dichos operadores compuestos comprenden unos operadores atómicos para renombrar los términos. También en este caso se aplica el tipo reetiquetar.

Considérese el ejemplo ilustrado en la figura 12. El patrón de la figura 12 y el mapeo muestran la situación de inicio:

```
map “/.../línea de dirección”
on la línea de Dirección del Grupo.
map “/.../país_nombre”
on País del Grupo.
```

Para hacer la semántica de este patrón más clara, se ejecuta el siguiente registro de cambios:

```
dropDifferentia (“Grupo”, “tiene”, “de”, “Línea de Dirección”)
dropDifferentia (“Grupo”, “tiene”, “de”, “País”)
defineDifferentia (“Persona”, “tiene”, “de”, “Línea de Dirección”)
defineDifferentia (“Persona”, “tiene”, “de”, “País”)
```

El resultado de este registro de cambios se muestra en la figura 17.
Y esto necesita el siguiente mapeo correspondiente:

```
map “/.../línea de dirección”
on Línea de dirección de la persona.
map “/.../país_nombre”
on País de Persona.
```

La solución proporcionada por la presente invención es en primer lugar la introducción de un operador compuesto para el patrón semántico que puede incorporarse en el registro de cambios. El operador compuesto define una secuencia específica de los operadores atómicos. En segundo lugar, este operador compuesto necesita tener un

operador equivalente que opere sobre los mapeos en el compromiso(s).
El nuevo operador se define de la siguiente manera:

```
renameTerm (X, Y)
  + introduceTerm (Y)
  + move (Zn, r'n, rn, X, Y)
  + dropConcept (X)
con (Zn, r'n, rn,) estando todas las relaciones asociadas con el término original
X.
```

que se ve de la siguiente manera en el ejemplo:

```
renameTerm ("Grupo", "Persona")
  + introduceTerm ("Persona")
  + move ("Línea de Dirección", "de", "tiene", "Grupo", "Persona")
  + move ("País", "de", "tiene", "Grupo", "Persona")
  + dropConcept ("Grupo")
```

En cuanto a los mapeos, el operador actúa sobre la parte de la ruta conceptual de los mapeos:

```
map "/.../línea de dirección"
on Línea de dirección del Grupo.
```

El operador expresa que esta ruta conceptual se ha renombrado, y realiza la operación:

```
map "/.../línea de dirección"
on Línea de dirección de la persona.
map "/.../país_nombre"
on País de la Persona.
```

Ha de observarse que la estructura de datos dentro de los ejemplos se especifica en XML, pero esto también puede ser una base de datos o cualquier otro formato de datos, tal como EDI, CSV o una base de datos SQL. Los métodos de la presente invención pueden extenderse a cualquier sistema de información y a cualquier formato de datos.

Los sistemas y los métodos de la presente invención también pueden aplicarse a los sistemas de ontología distribuidos.

Aunque la presente invención se ha ilustrado por referencia a las realizaciones específicas, será evidente para los expertos en la materia que la invención no está limitada a los detalles de las realizaciones ilustrativas anteriores, y que la presente invención puede realizarse con cambios y modificaciones diversos sin alejarse del alcance de las reivindicaciones. Las realizaciones presentes son, por lo tanto, para considerarse en todos los aspectos como ilustrativas y no restrictivas, indicándose el alcance de la invención por las reivindicaciones adjuntas más que por la descripción anterior, y todos los cambios que entren dentro del significado y del intervalo de equivalencia de las reivindicaciones están, por lo tanto, destinados a que estén abarcados por las mismas. En otras palabras, se contempla cubrir una y todas las modificaciones, variaciones o equivalentes que caigan dentro del alcance de las reivindicaciones y cuyos atributos esenciales se reivindican en esta solicitud de patente.

REIVINDICACIONES

1. Un método implementado por ordenador para modificar un mapeo desde al menos una ruta de aplicación de un sistema de datos usado por una aplicación informática a una ruta conceptual de un sistema de ontología enlazado con dicho sistema de datos, direccionando dicha ruta de aplicación una parte de la estructura de dicho sistema de datos y direccionando dicha ruta conceptual una parte de la estructura del sistema de ontología, comprendiendo dicho método las etapas de

- detectar un cambio en una parte de la estructura de dicho sistema de ontología que está direccionando una o más de dichas rutas conceptuales, siendo dicho cambio un cambio a nivel de base de lexon o a nivel de base de patrón, siendo dicha base de lexon un conjunto de lexones, definiendo cada lexon una relación entre dos términos en un contexto determinado, siendo dicha base de patrón un conjunto de patrones semánticos definidos por una selección de rutas conceptuales en dicha base de lexon y un conjunto de restricciones semánticas en dichas rutas conceptuales seleccionadas, describiéndose dicho cambio mediante una secuencia ordenada de primeros operadores que definen la información semántica sobre dicho cambio,
- definir uno o más segundos operadores que operan en dicho mapeo, siendo dicho uno o más segundos operadores equivalentes a dicho uno o más primeros operadores usados para describir dicho cambio en dicha parte de la estructura de dicho sistema de ontología,
- actualizar dicho mapeo con dicho uno o más segundos operadores equivalentes para reflejar el cambio en dicha parte de la estructura del sistema de ontología.

2. El método de acuerdo con la reivindicación 1, en el que dicho cambio detectado se almacena en un registro de cambios.

3. El método de acuerdo con la reivindicación 2, por el que se imponen una o más condiciones para aplicar dicho cambio en dicho sistema de ontología.

4. El método de acuerdo con cualquiera de las reivindicaciones 1 a 3, en el que dicho cambio comprende un reetiquetado de parte de la estructura del sistema de ontología.

5. El método de acuerdo con cualquiera de las reivindicaciones anteriores, en el que dicho cambio comprende añadir otra estructura a parte de la estructura del sistema de ontología, no siendo dicha otra estructura parte de la estructura del sistema de ontología antes del cambio.

6. El método de acuerdo con cualquiera de las reivindicaciones anteriores, en el que dicho cambio comprende insertar una estructura adicional a una parte de la estructura del sistema de ontología, no siendo dicha estructura adicional parte de la estructura del sistema de ontología antes del cambio.

7. El método de acuerdo con cualquiera de las reivindicaciones anteriores, en el que dicho cambio comprende eliminar parte de la estructura del sistema de ontología.

8. Un programa, ejecutable en un dispositivo programable que contiene instrucciones, que, cuando las ejecuta el dispositivo programable, realizan el método de acuerdo con cualquiera de las reivindicaciones anteriores.

9. Dispositivo para modificar un mapeo desde al menos una ruta de aplicación de un sistema de datos usado por una aplicación informática a una ruta conceptual de un sistema de ontología enlazado con dicho sistema de datos, direccionando dicha ruta de aplicación una parte de la estructura de dicho sistema de datos y direccionando dicha ruta conceptual una parte de la estructura del sistema de ontología, comprendiendo dicho dispositivo medios para detectar un cambio en una parte de la estructura de dicho sistema de ontología que está direccionando una o más de dichas rutas conceptuales, siendo dicho cambio un cambio a nivel de base de lexon o a nivel de base de patrón, siendo dicha base de lexon un conjunto de lexones, definiendo cada lexon una relación entre dos términos en un contexto determinado, siendo dicha base de patrón un conjunto de patrones semánticos definidos por una selección de rutas conceptuales en dicha base de lexon y un conjunto de restricciones semánticas en dichas rutas conceptuales seleccionadas, describiéndose dicho cambio mediante una secuencia ordenada de primeros operadores que definen la información semántica sobre dicho cambio, unos medios para definir uno o más segundos operadores que operan en dicho mapeo, siendo dicho uno o más segundos operadores equivalentes a dicho uno o más primeros operadores usados para describir dicho cambio en dicha parte de la estructura de dicho sistema de ontología y unos medios para actualizar dicho mapeo con dicho uno o más segundos operadores equivalentes para reflejar el cambio en dicha parte de la estructura del sistema de ontología.

10. Dispositivo de acuerdo con la reivindicación 9, en el que dicho dispositivo está dispuesto para identificar y almacenar información sobre la versión de dicho sistema de ontología.

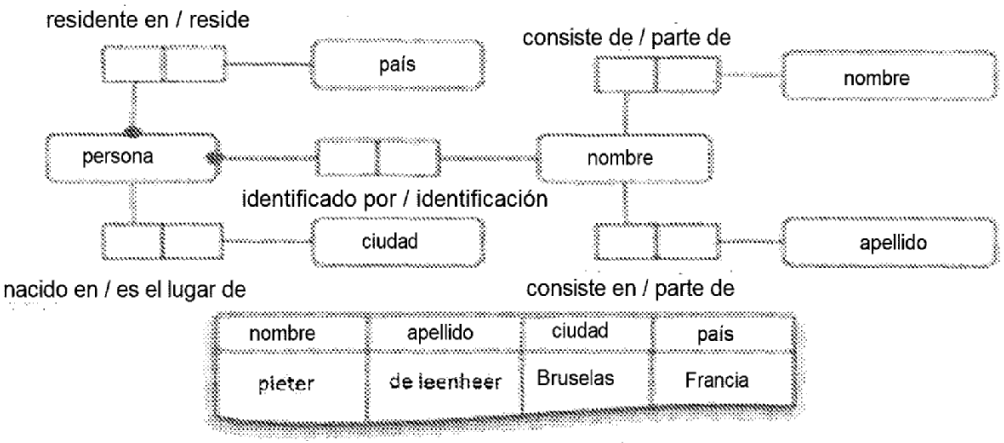
residente en / reside
país
consiste de / parte de
nombre
persona
nombre
identificado por / identificación
ciudad
apellido
nacido en / es el lugar de
consiste en / parte de
nombre
apellido
ciudad
país
pieter
de leenheer
Bruselas
Francia

Fig. 1

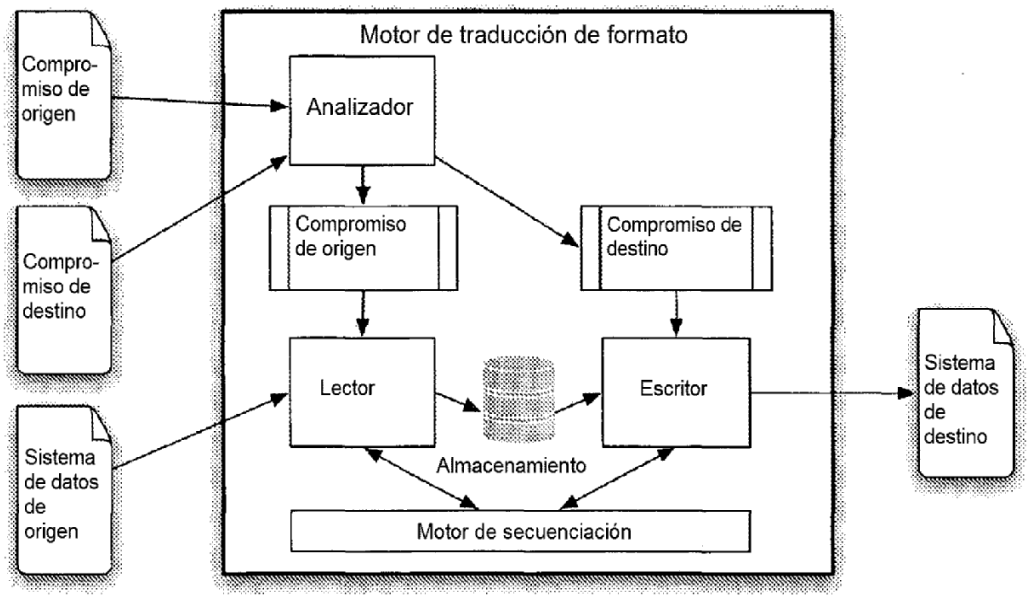
Motor de traducción de formato
Compro-miso de origen
Compro-miso de destino
Sistema de datos de origen
Analizador
Compromiso de origen
Compromiso de destino
Lector
Escritor
Almacenamiento
Motor de secuenciación
Sistema de datos de destino

Fig. 2

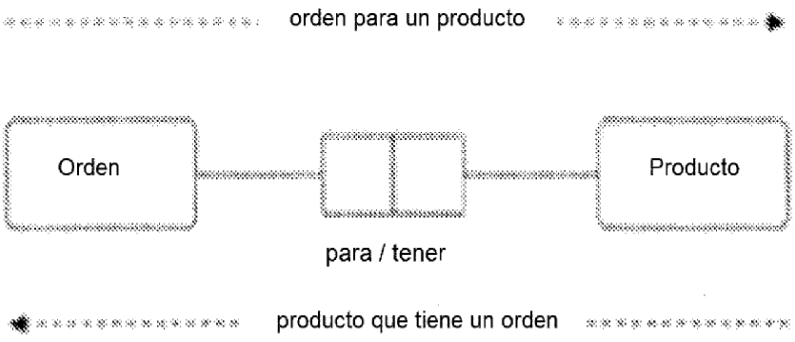
orden para un producto
Orden
Producto
para / tener
producto que tiene un orden

Fig. 3

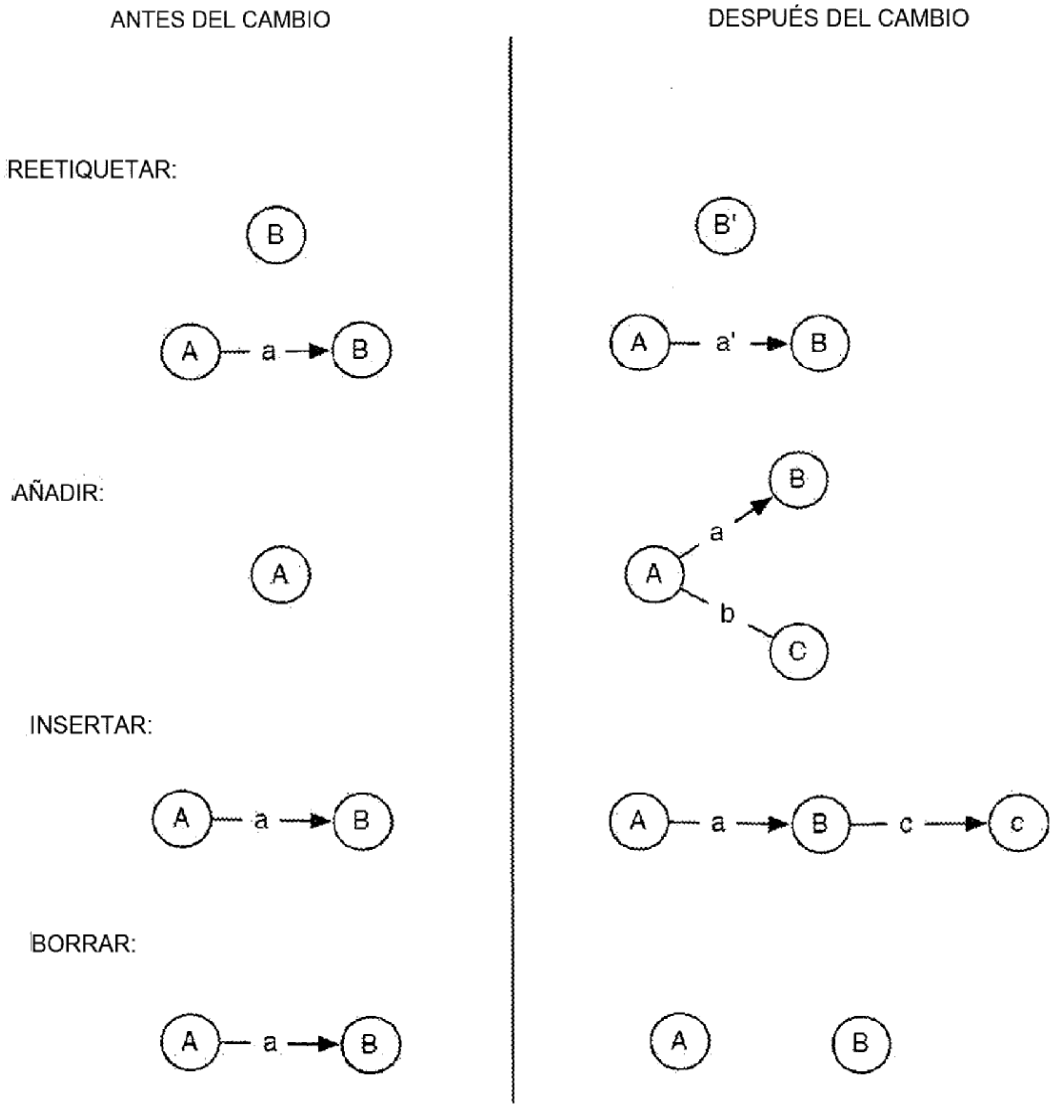
ANTES DEL CAMBIO
DESPUÉS DEL CAMBIO
REETIQUETAR:
B
A
a
B
B'
A
a'
B
AÑADIR:
A
B
a
A
b
C
INSERTAR:
A
a
B
A
a
B
c
c
BORRAR:
A
a
B
A
B

Fig. 4

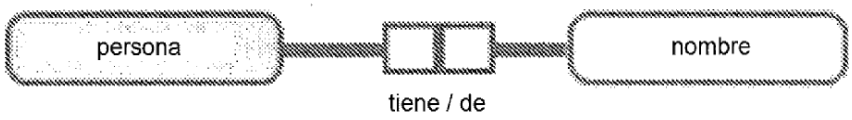

Fig. 5

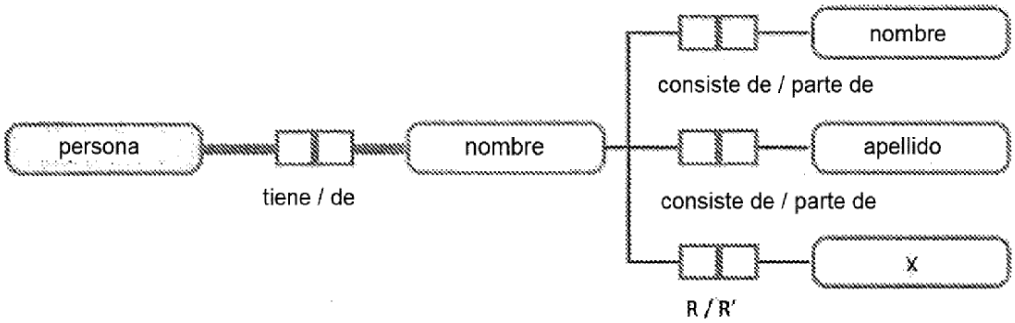

Fig. 6

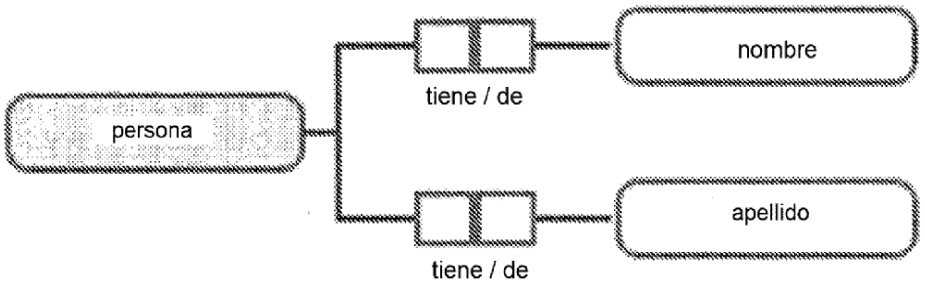

Fig. 7

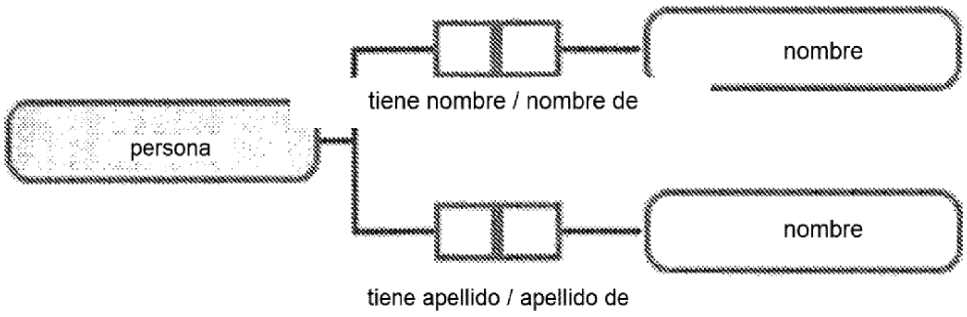

Fig. 8

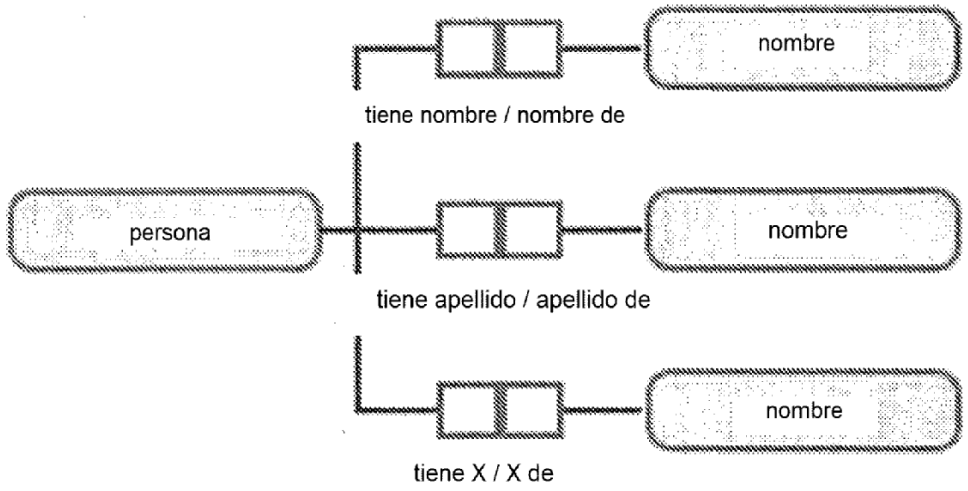

Fig. 9

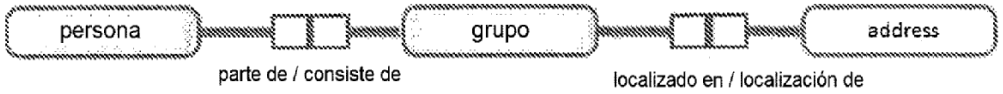

Fig. 10

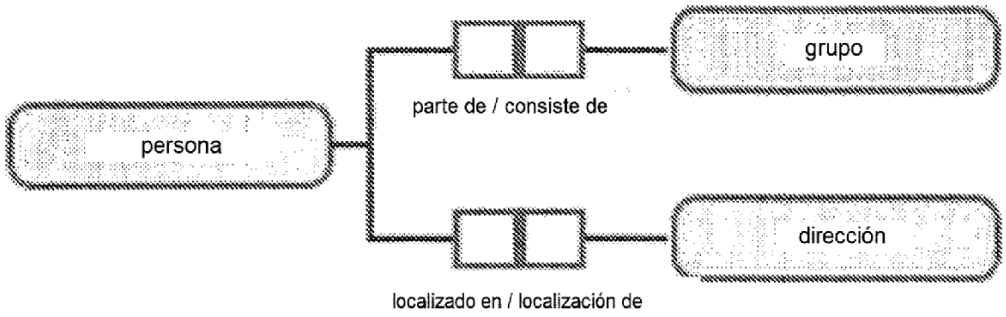

Fig. 11

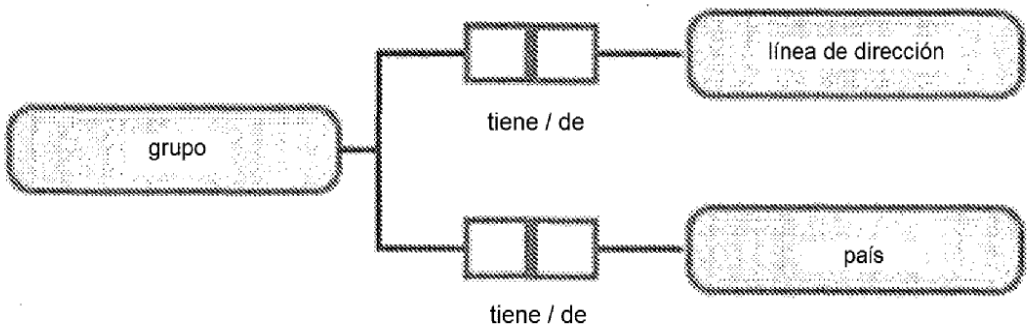

Fig. 12

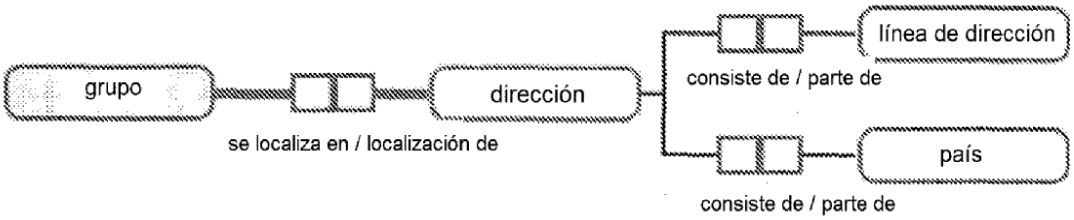

Fig. 13

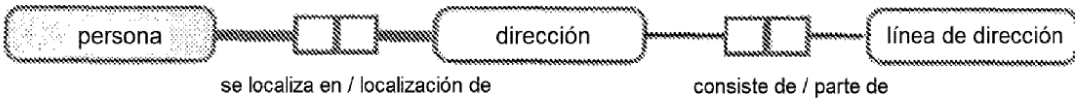

Fig. 14

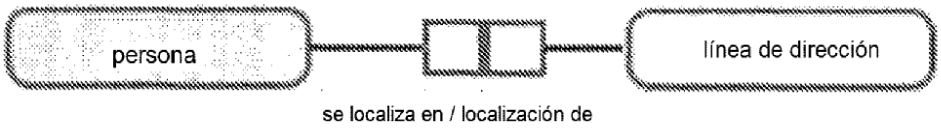

Fig. 15

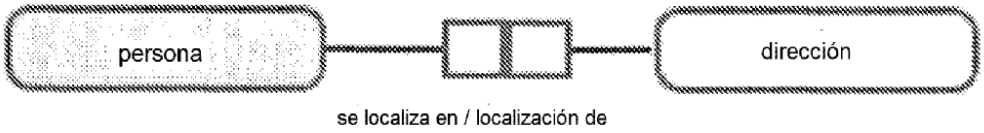

Fig. 16

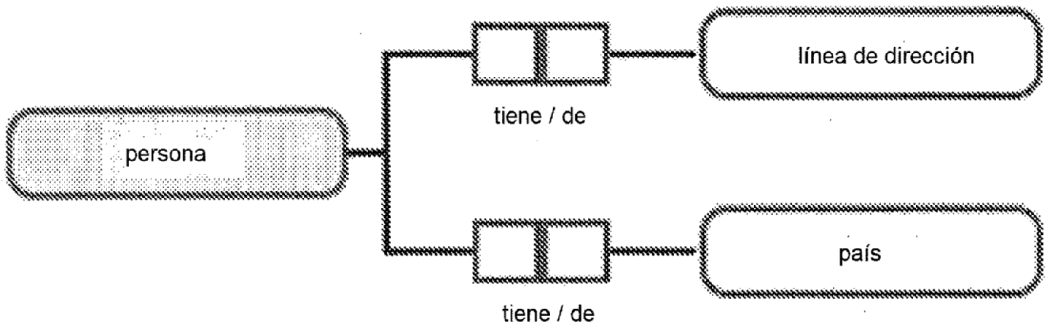

Fig. 17

REFERENCIAS CITADAS EN LA DESCRIPCIÓN

La lista de referencias citadas por el solicitante es, únicamente, para conveniencia del lector. No forma parte del documento de patente europea. Si bien se ha tenido gran cuidado al compilar las referencias, no pueden excluirse errores u omisiones y la OEP declina toda responsabilidad a este respecto.

Documentos de patente citados en la descripción

- EP 1970844 A **[0007] [0015]**
- US 2007162409 A **[0008]**
- US 2008071731 A **[0012]**

Literatura no patente citada en la descripción

- **A. DE MOOR et al.** Dogma-Mess : A Meaning Evolution Support System for Interorganizational Ontology Engineering. *Conceptual Structures : Inspiration and Application Lecture Notes in Computer Science,* 01 January 2006, vol. 4068, 189-202 **[0009]**
- **P. SPYNS et al.** Data Modelling versus Ontology engineering. *ACM Sigmod Record,* 31 December 2002, vol. 31 (4), 12-17 **[0010]**
- Object Role Modelling for Ontology Engineering in the DOGMA Framework. **P. SPYNS et al.** On the Move to Meaningful Internet Systems 2005. OTM Workshop Lecture Notes, 01 January 2005, vol. 3762, 710-719 **[0011]**
- **R. MEERSMAN.** Proc. of the International Symposium on Methodologies for Intelligent Systems. *ISMIS,* 1999, 30-45 **[0045]**