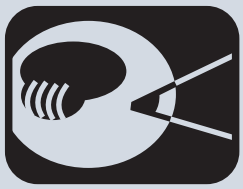
MODELO INTELIGENTE PARA BASES DE DATOS DISTRIBUIDAS



AUTOR
Ana C. Muñoz G.
Doctor (c) en Ciencias Aplicadas
Universidad de Los Andes
Docente Asistente
Instituto Universitario Tecnológico de Ejido Mérida
anamunoz@ula.ve
VENEZUELA

AUTOR
Jose Aguilar
Doctor en Ciencias Computacionales
Universidad Rene-Descartes, Francia
Docente Titular
Departamento de Computaciòn
Universidad de Los Andes
Aguliar@ula.ve
VENEZUELA

AUTOR
Rodrigo Martínez
Dr. En Informática
Docente Titular e Investigador
Universidad de Murcia
Rodrigo@um.es
ESPAÑA

Fecha de recepción del artículo:04 de Noviembre de 2005
Fecha de Aceptación del Artículo:18 de Noviembre de 2005
Articulo Tipo 1

RESUMEN.

En este trabajo trataremos el problema del diseño del "Modelo Inteligente para Sistemas de Bases de Datos Distribuidas". Particularmente, nos proponemos diseñar el modelo canónico a través del manejo ontológico de la información. Para esto se diseñan ontologías que permitirán describir una base de datos como un conjunto de términos representacionales de sus diferentes componentes. En estas ontologías, las definiciones asocian clases, relaciones, funciones, entre otras cosas, de entidades en el universo del discurso de las bases de datos, para describir el significado de las bases de datos, sus componentes, restricciones, etc. La razón de usar ontologías es que ellas definen conceptos y relaciones dentro de un marco taxonómico, cuya conceptualización está representada de una manera formal, legible y utilizable. En trabajos anteriores [14] se ha propuesto un modelo de referencia y una arquitectura para la integración de Bases de Datos en donde se plantea la necesidad de definir un modelo canónico. Como continuación de estos trabajos, en este artículo se describen las taxonomías ontológicas que componen el modelo de referencia para la integración de bases datos, y se diseña el Modelo Canónico usando dicha noción ontológica. De esta manera, se define el proceso de integración entre los diferentes tipos de bases de datos, estas bases de datos componentes pueden ser: Relacionales, Orientadas a Objeto, Difusas, Inteligentes y Multimedia. Así, el esquema ontológico describe los conceptos, operaciones y restricciones, tanto de las bases de datos componentes como de su proceso de integración. Además en este trabajo se muestra también los axiomas para cada una de los esquemas ontológicos utilizando lógica de predicado de primer orden.

PALABRAS CLAVES

Esquema ontológico
Modelo Canónico de Datos
Bases de Datos Distribuidas Inteligentes
Integración de Bases de Datos

ABSTRACT

In this abstract we will analyze-look at with the problem of the design of the "Intelligent Model for Distributed Database System". We particularly set out to design the canonical model through the ontological handling of the information. To do so, ontology is designed that allow the description of a database like a set of representative terms of their different components. In this ontology, the definitions associate classes, relations, functions, among other things, of organizations in the speech universe of the data bases, to describe their meaning, its components, restrictions, etc. The reason for using ontology is that it defines concepts and relations within a taxonomic frame, whose conceptualization is represented in a formal, legible and usable way. In previous works [14] a reference model and architecture for the integration of database the need to define an intelligent canonical model was proposed. Like continuation of these works, in this article the ontological taxonomies are described,

determining the component of the model of reference for the integration of database, and the Canonical Model is designed using this ontological notion. By doing so, the process of integration between the different types of database is defined. These component data bases can be: Relational, OO, Fuzzy, Intelligent and Multimedia. Thus, the ontological scheme describes the concepts, operations and restrictions, as well as, the component database and its process of integration. In this work there are also the axioms for each one of the ontological schemes using first-order predicate logic.

KEYWORDS

Ontological Scheme
Canonical data Model
Distributed Database Intelligent
Database Integration

INTRODUCCIÓN

La interoperabilidad entre diferentes sistemas de información es uno de los aspectos más críticos en la operación cotidiana de muchas organizaciones. En la última década esta preocupación se vio incrementada con la proliferación de diferentes bases de datos, con diferentes modelos de datos, que corren en diferentes plataformas. Los sistemas de bases de datos múltiples, también conocidos como bases de datos federadas, permiten tener disponible la información desde diferentes fuentes de información que pueden ser heterogéneas, distribuidas y autónomas. Una base de datos federada actúa como una aplicación front-end de múltiples componentes. La base de datos federada proporciona operaciones para el acceso a cada componente, manteniendo la consistencia de información entre las diversas fuentes y proporcionando un método de acceso uniforme a los servicios que cada componente ofrece.

La diversidad de lenguajes de programación y consulta, modelos de datos y métodos de integración, determinan diferentes estilos en la arquitectura de una base de datos federada, que varían desde un enfoque fuertemente acoplado al débilmente acoplado. En general, los sistemas fuertemente acoplados integran las diversas fuentes de información a través de un esquema conceptual global, normalmente denominado modelo canónico, proporcionando una vista uniforme de los diversos componentes a un alto nivel. El uso de un modelo canónico oculta las diferencias estructurales entre los diferentes componentes y da al usuario la ilusión de estar accesando una simple base de datos centralizada. Por otra parte, en los sistemas débilmente acoplados la integración de los componentes se basa en un lenguaje de acceso común que todos los componentes deben acordar, en el cual todas las funciones están estandarizadas.

En este trabajo trataremos el problema del diseño del "Modelo Canónico para Sistemas de Bases de Datos Distribuidas". Particularmente, nos proponemos diseñar el modelo canónico a través del manejo ontológico de la información. Para esto se diseñan ontologías que permitirán describir una base de datos como un conjunto de términos representacionales de sus diferentes componentes, ya sean datos, información o conocimiento. En estas ontologías, las definiciones asocian clases, relaciones, funciones, entre otras cosas, de entidades en el universo del discurso de las bases de datos, para describir el significado de las bases de datos, sus componentes, restricciones, etc.

La razón de usar ontologías es que ellas definen conceptos y relaciones dentro de un marco taxonómico, cuya conceptualización está representada de una manera formal, legible y utilizable. De esta forma, una ontología es un entendimiento común y compartido de un dominio, que puede comunicarse entre sistemas heterogéneos [12].

En trabajos anteriores [14] se ha propuesto un modelo de referencia y una arquitectura para la integración de Bases de Datos en donde se plantea la necesidad de definir un modelo canónico. Como continuación de estos trabajos, en este artículo se describen las taxonomías ontológicas que componen el modelo de referencia para la integración de bases datos, y se diseña el Modelo Canónico usando dicha noción ontológica. De esta manera, se define el proceso de integración entre los diferentes tipos de bases de datos y la resolución de conflictos a través de la ontología. En el futuro se diseñará el lenguaje de manipulación de las Bases de Datos Distribuidas Inteligentes usando la ontología que describe el modelo canónico. Para esto, se debe diseñar un mecanismo de inferencia que permitirá razonar durante los procesos de consulta y actualización a las Bases de Datos Distribuidas Inteligentes. También, a partir del mecanismo de inferencia se podrán hacer tareas de minería de datos, tales como generar patrones de acceso de usuarios al sistema para crear comunidades virtuales. Paso previo al diseño del Motor de Inferencia es traducir las ontologías a lógica de predicado de primer orden, para que a partir de ellas se puedan diseñar los mecanismos de consulta, actualización, y minería de datos para las Bases de Datos Distribuidas Inteligentes (es decir, las sentencias que describen el modelo canónico). En este trabajo se presenta dicha traducción.

La integración de bases de datos Federadas fuertemente acoplados han sido tratados en trabajos anteriores para bases de datos relacionales y objeto relacionales. Álvarez en su trabajo de tesis [23], presenta una propuesta de integración binaria que auxilia de manera semiautomática a la generación de una federación de bases de datos componentes, además de presentar un esquema auxiliar que contiene información para acceder los componentes locales a través de un lenguaje de consultas. En el trabajo Sistema cooperativo para la Integración de Fuentes Heterogéneas de Información y Almacenes de Datos de José Samos, de la Universidad Politécnica de Cataluña y la Universidad de Lleyda, presentan una arquitectura que integra el acceso integrado en tiempo real a las bases de datos utilizando un almacén de datos, utilizando como modelo canónico el modelo BLOOM. Estos trabajos utilizan la arquitectura para bases de datos federadas de Shet&Larson [20].

Este artículo muestra en una primera parte, el marco teórico en el que se basa el mismo, el cual abarca a las bases de datos distribuidas, específicamente las heterogéneas o federadas, su proceso de integración, así como las ontologías como

medio de integración. En la segunda parte se describe a través de esquemas ontológicos el proceso de integración, y se nombran los posibles tipos de bases de datos que pueden conformar una federación así como sus axiomas que definen las expresiones lógicas del proceso de integración; en el siguiente punto se describen los esquemas ontológicos de las bases de datos componentes descritos a través de sus conceptos, operaciones y restricciones.

Así, el aspecto fundamental de este trabajo es proponer un marco ontológico basado en sentencias de Lógica de Primer Orden (LPO) para la integración de una federación de bases de datos de diferentes tipos (Relacionales, Orientadas a Objeto, Multimedia, Difusas e Inteligentes).

1. MARCO TEÓRICO

1.1 BASES DE DATOS DISTRIBUIDAS

Las bases de datos distribuidas se refieren a la integración de necesidades de almacenamiento y procesamiento no locales en donde es necesario intercambiar información proveniente de diferentes sitios [1,2]. Los sistemas de bases de datos distribuidas integran sistemas de bases de datos diversos, para dar a los usuarios una visión global de la información disponible. La descentralización de la información promueve la heterogeneidad en su manejo. Esto hace que se dé en muchos niveles, desde la forma y significado de cada dato hasta el formato y el medio de almacenamiento que se elige para guardarlo. Desde el punto de vista funcional y de organización de datos, los sistemas de base de datos distribuidos están divididos en dos clases: Un SMBDD homogéneo que posee múltiples colecciones de datos e integra múltiples recursos de datos. Los sistemas homogéneos se parecen a un sistema centralizado, pero en lugar de almacenar todos los datos en un solo lugar los datos se distribuyen en varios sitios comunicados por la red. No existen usuarios locales y todos ellos accesan la base de datos a través de una interfaz global. Los sistemas heterogéneos se caracterizan por manejar diferentes SMBD en los nodos locales. Una subclase importante es la de los sistemas de manejo multi-bases de datos, llamadas también Bases de Datos Federadas, que integran información desde bases de datos heterogéneas, y presentan un acceso global a los usuarios, con métodos transparentes para usar la información total en el sistema. La principal característica es la autonomía que las bases de datos locales, también llamadas Bases de Datos Componentes, conservan para atender aplicaciones existentes. Para contar con una federación de Bases de Datos Componentes, es necesario proporcionar un mecanismo que sea capaz de conseguir un esquema global de Bases de Datos, el cual permita un acceso transparente a las diferentes Bases de Datos existentes, logrando así un enfoque global de los recursos de información de una organización [20].

La heterogeneidad en las bases de datos componentes se puede presentar en varios aspectos: hardware, software, modelado de datos y aspectos semánticos, entre otros. Un Sistema de Base de Datos Federado (SBDF) se clasifica como débilmente acoplado o fuertemente acoplado, basado en la idea de quién maneja la federación y cómo son integrados los componentes. Un SBDF es débilmente acoplado si la responsabilidad de crear y mantener la federación recae en el usuario, y no hay control por parte del sistema federado y sus administradores. Una federación es fuertemente acoplada cuando la federación y sus administradores son responsables de la creación y el mantenimiento de la misma, y participan activamente en el control de acceso de las Bases de Datos Componentes. Un sistema federado fuertemente acoplado puede ser de dos tipos: Con federación única, si permite la creación y gestión de un único esquema federado. Con múltiples federaciones, si permite la creación y gestión de múltiples esquemas federados [20]. Cada SBDF tiene una arquitectura de esquemas para superar las heterogeneidades sintácticas y semánticas, para ello consta de un cierto número de esquemas. Shet &Larson [20] proponen una arquitectura de esquemas para un SBDF, que son:

i) Esquema Local. Es el esquema conceptual de los Sistemas de Base de Datos Componentes que integran la Federación.
ii) Esquema Componente. Se convierten los esquemas conceptuales de las Bases de Datos Componentes a un modelo canónico, que es un modelo de datos común para todas las bases de datos que van a formar parte de la federación.
iii) Esquema de Exportación. En este esquema se establecen la parte de los esquemas componentes que se van a compartir así como su ubicación y el control de acceso.
iv) Esquema Federado, en este esquema se realiza la integración de los múltiples esquemas de exportación. Este esquema también incluye la información de la distribución de datos existentes.
v) Esquema Externo. En el que se definen los esquemas para cada usuario y/o aplicación.

1.2 MODELO CANÓNICO

La habilidad de representación de una Base de Datos viene dada por su modelo de datos. Un modelo de datos está compuesto por estructuras, operaciones y las restricciones en el uso de ellas. La habilidad de representación de un modelo de datos está compuesto por dos factores: expresividad y relativismo semántico [18].

Expresividad. La expresividad de un modelo de datos es el grado en el cual un modelo puede representar directamente los conceptos que la conforman, sin importar lo compleja que pueda ser esta. La expresividad está compuesta de una parte estructural y otra parte de comportamiento. La expresividad estructural es el poder de las estructuras del modelo para representar conceptos y ser interpretados como tales. La expresividad de Comportamiento, refleja el poder del modelo para representar el comportamiento de los conceptos.
Relativismo Semántico. El relativismo semántico de un modelo de datos es el poder de sus operaciones de derivar esquemas externos.

Cuando diferentes bases de datos interoperan forman una federación, y se debe elegir un modelo de datos como Modelo Canónico de Datos (MCD). El MCD es el elemento que procesa las consultas y actualizaciones que se le realizan a las bases de datos componentes de la federación. Así, siguiendo la

arquitectura de cinco niveles de Shet&Larson [20], cuando interoperan bases de datos preexistentes, debe adoptarse un MCD común a toda la federación. Los esquemas conceptuales e internos de las bases de datos componentes de la federación (modelos nativos) se integran en el MCD a través de les esquemas componentes. De esta manera, tendremos un esquema componente por cada tipo de base de datos que integra la federación, en nuestro caso para las bases de datos Relacionales, Orientadas a Objeto, Inteligentes, Multimedia, Temporales y Difusas. Cada esquema componente es filtrado en uno o más esquemas de exportación. De los esquemas de exportación de las bases de datos componentes se construye el esquema federado, este es resultado del proceso de integración. Finalmente, de un esquema federado se derivan diferentes esquemas externos. Así los esquemas propuestos en [20] están expresados en el MCD.

El uso de un MCD resuelve el problema de heterogeneidad sintáctica, consecuencia del uso de diferentes modelos de datos nativos. La heterogeneidad semántica, resultante de diferentes conceptualizaciones de las Bases de Datos Componentes, es resuelta en el proceso de integración de esquemas.

El modelo canónico de datos posee las siguientes características: Generalización: que es el proceso mediante el cual, a partir de dos o más entidades se construye una nueva entidad, eliminando sus diferencias y enfatizando sus similitudes; Agregación: que define nuevas entidades a partir de otras entidades; Clasificación: es la que permite agrupar entidades en clases, es decir construye una nueva entidad a partir de las características comunes de otras entidades; y Asociación: que define una nueva entidad a partir de las relaciones entre dos o mas entidades.

El MCD debe soportar la definición de nuevas operaciones y restricciones, debe permitir la implementación de operadores de integración, entre otras cosas [18].

Utilizaremos ontologías para representar nuestro MCD, ya que permiten integrar bases de datos utilizando inteligencia durante el proceso de conformación de la federación, así como el enriquecimiento semántico a través de la integración de las bases de datos con sus conceptos, operaciones y restricciones.

1.3 ONTOLOGíA

Una definición de Ontología hecha en términos de base de datos, es la que ofrece Weigand [13]:
"Una ontología es una base de datos que describe los conceptos del mundo de un dominio específico, algunas de sus propiedades y cómo estos conceptos se relacionan entre si".

El conocimiento representado dentro de una ontología es formalizado a través de cinco componentes:

Conceptos o clases: Son las ideas a formalizar. Son todas las ideas importantes relevantes para un determinado dominio de aplicación y pueden estar organizadas en taxonomias. Pueden ser descripción de objetos, tareas, funciones, acciones, estrategias, grupos, etc. Por ejemplo, la clase animal.
Relaciones: Representan las interacciones entre las clases, y están definidas como un subconjunto de un producto cartesiano. Por ejemplo, relación entre clase animal y clase alimento.

Funciones: Son casos especiales de relaciones, donde se generan elementos mediante el cálculo de una función. Por ejemplo: Precio_Objeto: Valor + Ganancia + IVA

Instancia: se usan para representar elementos o individuos en una ontología. Por ejemplo, la instancia perro de la clase animal.

Axiomas: Sirven para modelar sentencias que siempre van a ser ciertas. Se usan para representar conocimiento. Van a ser teoremas declarados sobre relaciones que deben cumplir los elementos de una ontología. Son utilizados para representar las propiedades que conceptos e instancias tienen que satisfacer. Por ejemplo: Si Clase animal es mamífero; la instancia perro es mamífero.

Se han propuesto clasificaciones de ontologías de acuerdo con el tipo de concepto a describir y su uso [7]. Así, se pueden encontrar los siguientes tipos de ontologías:

Terminológicas: especifican los términos que son utilizados para representar conocimiento en el universo del discurso. Suelen ser usadas para unificar vocabulario en un dominio determinado.

De Información: especifican la estructura de almacenamiento de bases de datos. Ofrecen un marco para el almacenamiento estandarizado de información.

De Modelado del Conocimiento: Especifican conceptos vinculados al conocimiento. Contienen una estructura interna rica y suelen estar ajustadas al uso particular del conocimiento que describen.

Existen otras clasificaciones de ontologías atendiendo a diversos criterios, una de ellas es atendiendo a su alcance de aplicabilidad. Según este criterio, se pueden distinguir los siguientes tipos de ontologías [7]:

Ontologías de dominio: Estas ontologías son específicas para un dominio en concreto. Por ejemplo, una ontología en el dominio de los Sistemas de Información Geográfica.

Ontologías de tareas: Estas ontologías representan las tareas que son susceptibles de realizar en un dominio en concreto. Por ejemplo, una ontología para las tareas de mantenimiento preventivo.

Ontologías Generales: Van a representar los datos generales y no específicos de un dominio. Por ejemplo, ontologías sobre el tiempo, el espacio, etc.

1.4 METODOLOGíA PARA LA CONSTRUCCIóN DE ONTOLOGíAS

Existen metodologías para la construcción de ontologías. Las metodologías de diseño pueden construir una ontología utilizando diferentes técnicas: las que construyen la ontología aplicando un proceso bottom-up, que comienzan desde conceptos simples y los agrupan en conceptos más generales; y aquellas que construyen la ontología aplicando un proceso top-down. Otras aplican procesos mixtos [7]. Presentamos algunas de ellas, por ejemplo el método usado para construir la ontología Cyc KB consiste de tres fases: La primera fase consiste en la codificación manual de artículos y piezas de conocimiento, en las cuales el conocimiento común que está implícito en diferentes fuentes se extrae a mano. La segunda y tercera fase consiste en adquirir nuevo conocimiento usando lenguaje natural o herramientas de máquinas de aprendizaje.

En [7] proponen una metodología inspirada en el desarrollo de sistemas basados en conocimiento usando lógica de primer orden. Ellos proponen primero, identificar intuitivamente los principales escenarios (las posibles aplicaciones en las que la ontología será utilizada). Luego utilizan un conjunto de preguntas en lenguaje natural, para determinar el alcance de la ontología. Estas preguntas y sus respuestas son utilizadas para extraer los principales conceptos y sus propiedades, relaciones y axiomas de la ontología. Tales componentes ontológicos se expresan formalmente en lógica de primer orden.

Para el desarrollo de nuestra ontología utilizaremos en una primera parte, la identificación de las principales piezas de conocimiento, como son los tipos de bases de datos que conforman la federación. Luego se establecerán las principales características de la ontología: cómo se realiza la integración y sus restricciones, los conceptos y sus propiedades, las relaciones y axiomas tanto de cada tipo de base de datos como del proceso de integración. Se establece la estructura de la ontología a través de jerarquías de tipo taxonómico. Todo esto utilizando una técnica top-down.

1.5 LENGUAJES DE REPRESENTACIóN DE ONTOLOGíAS

Los lenguajes de las ontologías están basados en lógica de primer orden (por ejemplo, KIF (Knowledge Interchange Format), ONTOLINGUA y OCML (Operational Conceptual Modelling Language, y FLogic) [7]. KIF permite la representación de conceptos, taxonomías de conceptos, relaciones n-arias, funciones, axiomas, instancias y procedimientos, pero no existe soporte de razonamiento. OIL (Ontology Inference Layer) [7] lenguaje que unifica tres aspectos: semánticas formales y soporte de razonamiento provisto por lógicas de descripción, primitivas de modelado y un estándar para intercambiar notaciones sintácticas.

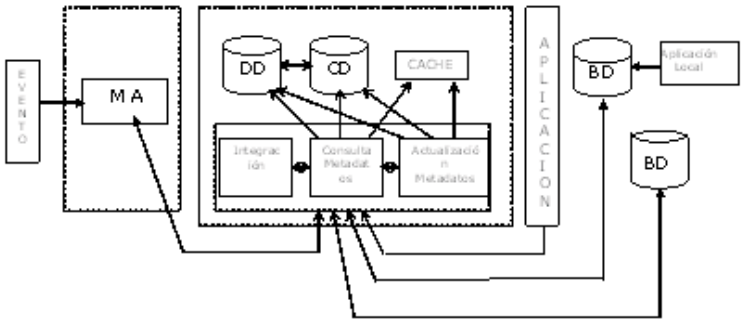
Por otro lado, ODE es una herramienta para la construcción de ontologías que interactúa con el usuario, proporcionando diferentes representaciones intermedias de ontologías que son independientes del lenguaje en que la ontología es implementada. SHOE es un lenguaje de representación de conocimiento XML-compatible para Web. El lenguaje ontológico Web OWL (Web Ontology Language) proporciona un lenguaje que formaliza un dominio definiendo clases y sus propiedades, propiedades individuales y aserciones sobre ellas, y razonamiento acerca de las clases e individuos al grado permitido por la semántica formal del mismo. El OWL facilita más interpretabilidad en la Web que el soportado por XML, RDF, y RDF Schema proporcionando vocabulario adicional junto con una semántica formal [4].

Finalmente en los últimos años surgió una nueva generación de ambientes para desarrollo de ontologías. Están construidas como aplicaciones robustas que proporcionan soporte tecnológico en la mayoría de las actividades del ciclo de vida de la ontología. Son extensibles, con arquitectura basada en componentes, donde los nuevos módulos pueden ser fácilmente agregados para proporcionar más funcionalidad independiente. Además, los modelos de conocimiento bajo este ambiente son independientes del lenguaje. Entre estos ambientes se encuentran Protégé 2000, WebODE, OntoEdit y Kaon [7].

2. PROPUESTA ORIGINAL EN DE BASES DE DATOS DISTRIBUIDAS INTELIGENTES

Ahora bien las Bases de Datos Distribuidas Inteligentes están basadas en dos grandes áreas: el área de distribución, como son las bases de datos federadas; y el área de bases de datos inteligentes, como son las bases de datos activas, deductivas, difusas y basadas en conocimiento. Además, ellas incluyen las convencionales como las relacionales, y otras más recientes como las orientadas a objeto, multimedia y temporales. En la integración de las bases de datos federadas se incorpora la inteligencia a través del puente propuesto en [14] ver figura 1, el cual requiere de un modelo canónico para realizar dicha tarea. La inteligencia se introduce por la capacidad del puente de representar conocimiento, aprender y razonar, tareas que le corresponden a la máquina de aprendizaje (MA), los componentes del puente son:

Figura 1. Arquitectura del puente integrador



DD (Diccionario de Datos), es un almacén que contiene los diferentes tipos de datos de la federación (metadatos).
CD (Control de Datos), donde se definen las autorizaciones de acceso a las bases de datos.
BD (Bases de Datos), son las bases de datos locales.
MA (Máquinas de Aprendizaje), donde se realiza el proceso de razonamiento (contiene el Motor de Inferencia)

Consulta Metadatos: En éste módulo se procesan los requerimientos de búsquedas en la federación.
Actualización Metadatos: En ella se realizan todas las tareas de actualización en la federación, que involucra el manejo de la coherencia en las diferentes bases de datos, entre otras cosas.
Cache: guarda la información más recientemente accesada en la federación.
Integración: Resuelve los conflictos de heterogeneidad sintáctica, semántica y de seguridad entre las diferentes bases de datos locales.

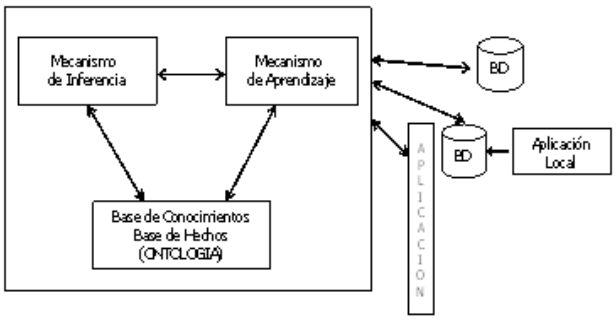
Normalmente, el modelo canónico es una metabase de datos resultante del proceso de integración de las bases de datos del sistema federado. Ahora bien, proponemos fusionar los módulos funcionales del puente en el modelo canónico, es decir la Máquina de Aprendizajes y los Módulos de Integración, Actualización y Consulta. Para esto, basaremos el diseño del Modelo Canónico en el Modelo de Referencia presentado en [20] y en el uso de un marco ontológico que lo describe a él y a sus funciones.

3. DISEÑO DE UN MODELO INTELIGENTE DE INTEGRACIÓN PARA BASES DE DATOS FEDERADAS

El diseño de nuestro modelo canónico se basará en una Ontología que identifique los tipos de base de datos a integrar. Esta ontología será el resultado de la unión de ontologías que describen a cada una de las bases de datos a integrar, así como de elementos propios del proceso de integración. En la siguiente figura se muestra la evolución del puente anterior a un Modelo Canónico Inteligente que utiliza la ontología (modelada en una Base de Conocimientos y de Hechos), y que además posee mecanismos de razonamiento y de aprendizaje para llevar a cabo el proceso de integración.

En la figura 2 se muestran éstos módulos y sus relaciones para llevar a cabo la Integración Inteligente en Bases de Datos Federadas.

Figura 2. Modelo Inteligente de Integración para Bases de Datos Federadas



A continuación se describen los esquemas ontológicos para el proceso de integración de una base de datos federada así como la de cada una de las bases de datos que pueden conformar una federación

3.1 BASES DE DATOS FEDERADAS

Las Bases de Datos Federadas integran información desde bases de datos heterogéneas locales en un ambiente distribuido, y presentan un acceso global a los usuarios, para usar la información total del sistema. La principal característica es la autonomía que las bases de datos locales, o Bases de Datos Componentes, conservan para atender las aplicaciones. Para contar con una federación de Bases de Datos Componentes, es necesario proporcionar un mecanismo de integración que sea capaz de conseguir un esquema global de Bases de Datos, logrando un enfoque global de los recursos de información de una organización. Esto se obtiene a través del modelo Canónico, definido ahora por nosotros como una ontología que contiene los conceptos, operaciones y restricciones necesarios para poder realizar la integración inteligente de las bases de datos Componentes.

3.1.1 Conceptos para Bases de Datos Federadas

Las Bases de Datos Federadas son bases de datos componentes y tienen operaciones de integración y restricciones de integración. Las Bases de Datos Componentes son aquellas bases de datos que van a conformar una federación. En nuestro caso, estas bases de datos componentes pueden ser: Bases de Datos Relacionales, Bases de Datos Orientadas a Objeto, Bases de Datos Multimedia, Bases de Datos Difusas, Bases de Datos Inteligentes; también una base de datos componente puede ser otra base de datos federada. Cada una de estas bases de datos componentes tienen sus conceptos, operaciones y restricciones.

En la figura 3 se muestra el esquema ontológico que describe los conceptos de las bases de datos federadas, en esta figura se pueden ver los enlaces a los siguientes esquemas ontológicos que se describen a lo largo de todo el artículo.

3.1.2 Axiomas para los conceptos de Bases de Datos Federadas

A continuación se describe el comportamiento del esquema ontológico anterior a través de los axiomas. Estos se utilizan para definir la ontología por medio de expresiones lógicas. Cada axioma definido incluye: su descripción en lenguaje natural, y la expresión lógica que describe formalmente el axioma.

Figura 3. Esquema Ontológico de los conceptos Para Bases de Datos Federadas

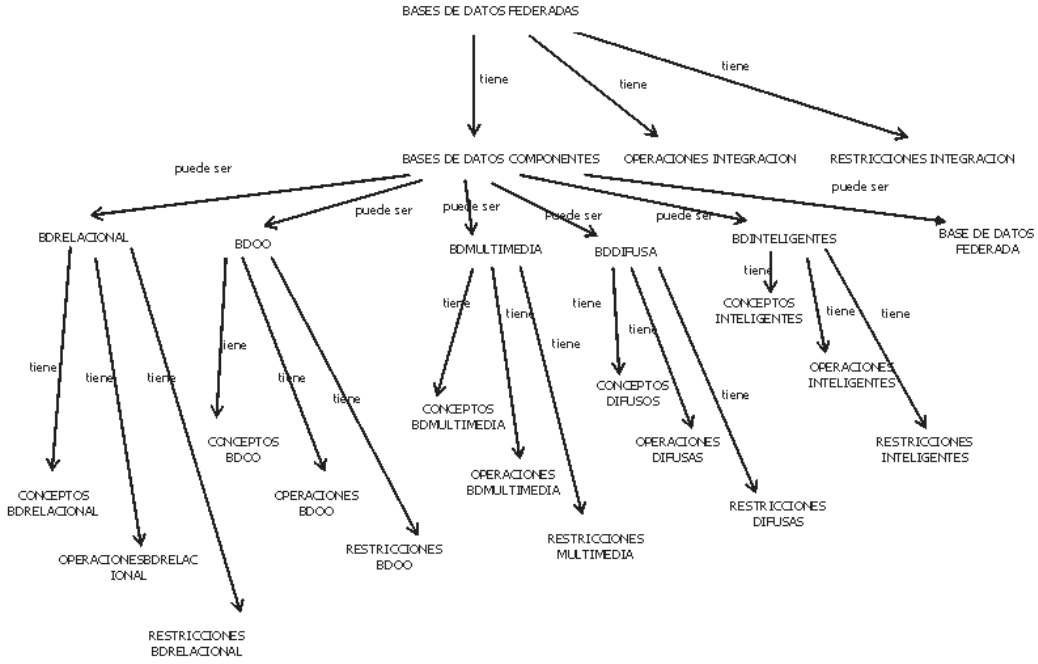


Tabla 1. Expresiones Lógicas para los Axiomas

Sentencia	LPO
Una base de datos federada tiene base de datos componentes y tiene operaciones de integración y restricciones de integración	∀ x BDFederada(x) => Tiene (x,BDComponente) ∧ Tiene (x,OperacionIntegracion) ∧ Tiene (x,RestriccionIntegracion)
Las bases de datos componentes pueden ser bases de datos relacionales, bases de datos orientadas a objeto, bases de datos multimedia, bases de datos difusas, bases de datos multimedia, bases de datos inteligentes y bases de datos federada	∀ x BDComponente(x) => PuedeSer(x,BDRelacional) ∨ PuedeSer (x,BDOO) ∨ PuedeSer (x, BDMultimedia) ∨ PuedeSer (x, BDDifusa) ∨ PuedeSer (x, BDInteligentes) ∨ PuedeSer (x, BDComponente)
Las BDRelacionales tienen Conceptos, Operaciones, Restricciones	∀ x BDRelacional(x) => Tiene(x,ConceptosR) ∧ Tiene(x, OperacionesR) ∧ Tiene(x, RestriccionesR)
Las BDOO tienen Conceptos, Operaciones, Restricciones	∀ x BDOO(x) => Tiene(x,ConceptosOO) ∧ Tiene(x, OperacionesOO) ∧ Tiene(x, RestriccionesOO)
Las BDMultimedia tienen Conceptos, Operaciones, Restricciones	∀ x BDMultimedia(x) => Tiene(x,ConceptosMM) ∧ Tiene(x, OperacionesMM) ∧ Tiene(x, RestriccionesMM)
Las BDDifusas tienen Conceptos, Operaciones, Restricciones	∀ x BDDifusas(x) => Tiene(x,ConceptosDF) ∧ Tiene(x, OperacionesDF) ∧ Tiene(x, RestriccionesDF)
Las BDInteligentes tienen Conceptos, Operaciones, Restricciones	∀ x BDInteligentes(x) => Tiene(x,ConceptosInt) ∧ Tiene(x, OperacionesInt) ∧ Tiene(x, RestriccionesInt)

3.2 OPERACIONES DE INTEGRACION LAS BASES DE DATOS FEDERADAS

Nosotros usaremos las operaciones de integración de acuerdo a Batini y Lenzerini [3], las cuales se realizan en fases. A continuación se describen las características de estas fases.
Preintegración. En esta fase se deciden las bases de datos a integrar, el orden de integración, que puede ser binario cuando se integran dos esquemas a la vez, y n-arias cuando se integran n esquemas a la vez; y las partes de las bases de datos a integrar. También se establecen las políticas de integración en cuanto a las restricciones de acceso y prioridad en el acceso a las Bases de Datos Componentes. Se realiza un proceso de negociación, ya sea para formar una nueva federación o para incorporar una base de datos componente a un sistema de base de datos federados existente.

Comparación de los esquemas. Las bases de datos se comparan y analizan para determinar la correspondencia entre conceptos y detectar los posibles conflictos. Una vez que se detectan los conflictos, se incorporan a la base de conocimientos de conflictos para manejar sus soluciones a través de un sistema de reglas.

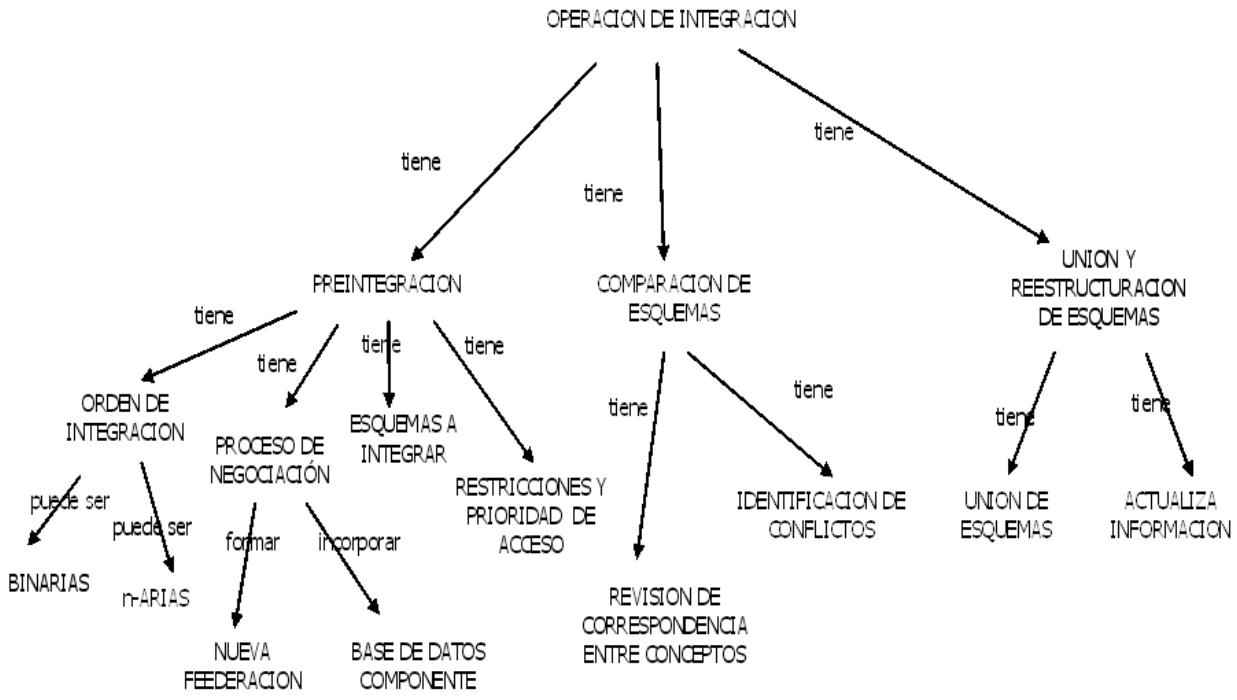
Unión y Reestructuración. Una vez detectados los conflictos, se resuelven de manera tal que la unión de los diferentes esquemas de las bases de datos componentes sea posible. La meta de esta actividad es conformar o alinear esquemas para hacerlos compatibles para su integración. Tiene las operaciones como: renombrado, los conceptos atómicos se transforman uno en otro, por ej, transformar entidades/ relaciones/atributos, eliminación de relaciones redundantes, crear jerarquía de generalización, definición de subtipos y supertipos. En la figura 4 se muestra el esquema ontológico que describe las operaciones de integración de bases de datos federadas.

Los Axiomas para las operaciones de integración de las Bases de Datos Federadas son:

Tabla 2. Axiomas para Operaciones de Integración

Sentencia	LPO
La operación de integración tiene la fase de preintegración, comparación de esquemas y conformación del modelo canónico	∀ x OperaciónIntegración(x) => Tiene (x,Preintegración) ∧ Tiene(x,ComparacióndeEsquemas) ∧ Tiene (x,UnionyReestructuracionEsquemas)
La Preintegración tiene orden de integración, tiene un proceso de negociación y tiene esquema a integrar y tiene restricciones y prioridad de acceso	∀ Preintegración(x) =>Tiene(x, OrdenIntegracion) ∧ Tiene(x,ProcesoNegociación) ∧ Tiene(x,EsquemasaIntegrar) ∧ Tiene(x, RestriccionesdeAcceso) ∧ Tiene(x, PrioridaddeAcceso)
El Orden de integración de la Base de Datos puede ser binario o n-ario	∀ x OrdenIntegracion(x) => Puedeser(x,IntegracionBinario) ∨ PuedeSer(x,IntegracionN-ario)
En el Proceso de negociación se forma una nueva federación o se incorpora una base de datos componente	∀ x ProcesoNegociación(x) => Forma(x,NuevaBDFederada) ∨ Incorpora(x,BDComponente)
En la comparación de esquemas tiene revisar correspondencia entre conceptos y tiene identificación de conflictos	∀ x ComparacionEsquemas(x) => Tiene (x,RevisarCorrespondenciaentreConceptos) ∧ Tiene(x,IdentificarConflictosIntegracion)
Un orden de integración binario es aquel integra dos esquemas a la vez	∀ x IntegracionBinaria(x) => Integra(x,DosEsquemas)
El orden de integración n-ario es el que integra n esquemas a la vez	∀ x IntegracionBinaria(x) => Integra(x,NEsquemas)
Las restricciones de acceso son las autorizaciones para acceder a las bases de datos componentes que conformaran la federación	∀ x RestriccionesdeAcceso(x) => Autoriza(x,AccesosaBDComponente)
La prioridad de acceso establece el orden de acceso a las bases de datos componentes	∀ x PrioridaddeAcceso(x) => Establece(x,OrdendeAcceso)
La unión y reestructuración de esquemas tiene la unión de esquemas y la actualización de información en el modelo	∀ x UnionyReestructuracionEsquemas(x) => Tiene(x,UnionEsquemas) ∧ Tiene(x,ActualizacionInformación)

Figura 4. Esquema Ontológico de Operaciones de Integración para Bases de Datos Federadas



3.3 RESTRICCIONES DE INTEGRACIóN EN BASES DE DATOS FEDERADAS

En la integración de las bases de datos pueden presentarse los siguientes tipos de conflictos:

Conflictos de Esquemas.

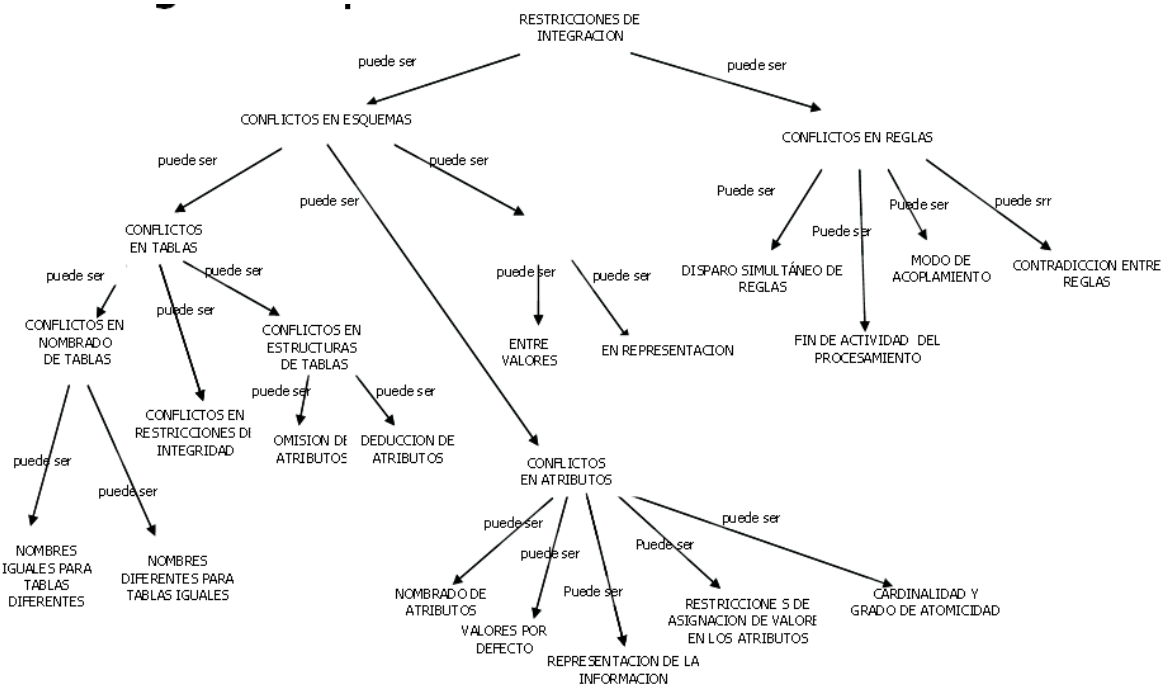
- *Conflictos en Tablas.*
  - Conflictos en el Nombrado de tablas: pueden surgir cuando existen nombres diferentes para tablas equivalentes o nombres iguales para tablas diferentes.
  - Conflictos en la Estructura de las tablas, objetos y elementos multimedia: cuando hay atributos, objetos o elementos multimedia que se omiten porque no se consideran representativos para la integración o atributos, objetos o elementos multimedia implícitos que se deducen de los existentes necesarios para la integración.
  - Conflictos en las Restricciones de Integridad, diferencia que surgen con respecto a la selección de claves primarias y ajenas.

- *Conflictos de Atributos.*
  - Conflictos en nombrado de Atributos, pueden surgir cuando existen nombres diferentes para atributos equivalentes o nombres iguales para atributos diferentes.
  - Conflictos en Valores por Defecto, es una definición implícita de los valores deducidos por defecto
  - Conflictos por Restricciones de Asignación de Valores a los Atributos, pueden ser conflictos en los tipos de datos y conflictos en las restricciones de dominio.
  - Conflictos por la Cardinalidad y Grado de Atomicidad, el grado de detalle de cada atributo puede ser distinto en cada aplicación.
  - Conflictos en la Representación de la Información, donde un mismo concepto puede representar un atributo y una entidad por ejemplo.

- *Conflictos de Datos.*
  - Conflictos entre los valores, cuando instancias equivalentes no tienen los mismos valores debido a que los datos obtenidos son incorrectos o son obsoletos.
  - Diferencias en la representación, vienen dadas por situaciones de contexto y cultura organizacional. Estas diferencias vienen dadas por: notaciones diferentes, donde un dato se representa de diferentes maneras (por Ej.: calificaciones numéricas y letras), unidades distintas (por Ej., sistemas de medición inglés y el internacional), y diferencias en la representación (cuando existen diversas maneras para representar un dato, por Ej.: para describir un estado Caracas y CCS o Mérida y MRD)

- ***Conflictos en Reglas:***
  - Disparo Simultáneo de Reglas, cuando un evento o consulta tiene diferentes reglas asociadas y el sistema permite que solo una regla se active
  - Modo de Acoplamiento, Acoplamiento entre la detección del evento y la evaluación de la condición, y luego entre la evaluación de la condición y la ejecución de la acción
  - Terminación de la actividad del procesamiento de las Reglas.
  - Contradicción entre reglas, cuando el evento o la consulta dispara una regla que es la negación del evento de la regla previamente disparada

Figura 5. Esquema Ontológico de las Restricciones de Integración para Bases de Datos Federadas.



Los Axiomas de las restricciones de integración para las Bases de Datos Federadas son:

Tabla 3. Axiomas de las restricciones de integración

Sentencia	LPO
Las restricciones de integración pueden ser conflictos en esquemas o conflictos en reglas	∀ x RestriccionIntegracion (x) => PuedeSer(x,ConflictoEsquema) V PuedeSer(x,ConflictoReglas)
Los conflictos en esquema pueden ser conflictos en tablas o conflictos en atributos o conflictos en datos	∀ x ConflictodeEsquema(x) => PuedeSer(x,ConflictoTabla) V PuedeSer(x,ConflictoAtributo) V (x,ConflictoDatos)
Los conflictos en tablas pueden ser en nombrado de tablas, en estructura de tabla, objeto o elemento multimedia o en restricciones de integridad	∀ x ConflictoTabla(x) => PuedeSer (x,ConflictoNombreTabla) V PuedeSer (x, ConflictoEstructuraTabla) V PuedeSer(x,ConflictoEstructuraObjeto) V PuedeSer (x, ConflictoEstructuraMultimedia) V PuedeSer(x,ConflictoRestriccionIntegridad)
El conflicto en nombrado de tablas surge cuando existen nombres diferentes para tablas iguales o nombres iguales para tablas diferentes	∀ x ConflictoNombreTabla(x) => NombreTablaDiferente(x,TablasEquivalentes) V NombreTablaIgual(x,TablasDiferentes)
El conflicto en estructura de tabla ocurre cuando hay atributos que se omiten o cuando hay atributos que se deducen	∀ x ConflictoEstructuraTabla (x) => PuedeSer(x,AtributosTOmitidos) V PuedeSer(x,AtributosTDeducidos)
El conflicto en estructura de Objeto ocurre cuando hay atributos del objeto que se omiten o cuando hay atributos del objeto que se deducen	∀ x ConflictoEstructuraObjeto(x) => PuedeSer(x,AtributosObOmitidos) V PuedeSer(x,AtributosObDeducidos)
El conflicto en estructura multimedia ocurre cuando hay atributos del objeto multimedia que se omiten o cuando hay atributos del objeto multimedia que se deducen	∀ x ConflictoEstructuraMultimedia => PuedeSer(x,AtributosMMOmitidos) V PuedeSer(x,AtributosMMDeducidos)
Los conflictos en atributos pueden ser conflictos en nombre de atributo o conflictos en valores por defecto o conflictos de restricciones de asignación de valores de los atributos o conflictos de cardinalidad o conflictos en la representación de la información	∀ x ConflictoAtributo(x) => PuedeSer(x,ConflictoNombreAtributo) V PuedeSer(x,ConflictoValorporDefecto) V PuedeSer (x,ConflictoRestriccióndeAsignacióndeValores) V PuedeSer (x,ConflictoCardinalidad) V PuedeSer (x,ConflictoRepresentacióndeInformacion)
Los conflictos en nombrado de Atributos surgen cuando existen nombres diferentes para atributos equivalentes o nombres iguales para atributos diferentes	∀ x ConflictoNombreAtributo(x) => TieneNombreDiferentes(x,AtributosEquivalentes) V TienenNombresIguales(x,AtributosDiferentes)
Los conflictos en valores por defecto es la definición de los valores deducidos por defecto	∀ x ConflictoValorporDefecto(x) => Tiene(x,Definiciónde ValoresDeducidos)
Los conflictos por Restricciones de Asignación de Valores a los Atributos, pueden ser conflictos en los tipos de datos y conflictos en las restricciones de dominio.	∀ x ConflictoRestriccióndeAsignacióndeValores(x) => PuedeSer(x,ConflictosenTipodeDatos) V (x,ConflictoserRestriccionesdeDominio)
Un conflicto de cardinalidad es la diferencia de detalles de los atributos	∀ x ConflictoCardinalidad(x) => Tiene(x,DiferenteNivelde Representacióndeatributos)
Los Conflicto en la representación de información son las diferentes dominios que representa un atributo	∀ x ConflictoRepresentacióndeInformacion(x) => Tiene(x,DominiosDiferentes)
Los conflictos en datos pueden ser conflicto entre valores o conflicto de diferencias en la representación	∀ x ConflictoDatos(x) => PuedeSer (x,ConflictoentreValores) V PuedeSer (x,ConflictodeDiferenciadeRepresentación)
Un conflicto entre valores surge cuando instancias iguales tienen valores diferentes	∀ x ConflictoentreValores(x) => Tienen(x,InstanciasIgualesdeDatos) Λ Tiene(x,ValoresDiferentesdeDatos)
Las diferencias de representación tienen varias representaciones diferentes para un mismo dato	∀ x ConflictodeDiferenciadeRepresentación(x) => Tiene(x,RepresentacionDiferentesparaIgualDato)
Un conflicto en regla puede ser un disparo simultaneo de reglas, o puede ser un conflicto en el modo de acoplamiento o puede ser conflicto en el fin del procesamiento de reglas, o puede ser una contradicción entre reglas	∀ x ConflictoenReglas(x) => PuedeSer (x,DisparoSimultaneoReglas) V PuedeSer (x,ConflictoenelMododeAcoplamiento) V PuedeSer (x,FinProcesamiento) V PuedeSer (x,ContradiccionEntreReglas)
Un disparo simultaneo de reglas es cuando un evento dispara mas de una regla	∀ x DisparoSimultaneoReglas(x) => EsUn(x,DisparamasdeunaRegla)
Los conflictos en el modo de acoplamiento es cuando una regla tiene desacoplamiento entre el evento y la condición y entre la condición y la acción	∀ x ConflictoenelMododeAcoplamiento => Tiene(x,DesacoplamientoentreEventoyCondición) Λ Tiene(x, DesacoplamientoentreCondiciónyAccion)
La contradicción entre reglas existe cuando el evento dispara una regla que es la negación del evento de la regla disparada	∀ x ContradiccionEntreReglas(x) => Tiene(x,Evento) Λ Tiene (x,¬Evento)

4. TAXONOMIA DE LAS BASES DE DATOS COMPONENTES

A continuación se describen cada una de las taxonomías que describen las bases de datos que intervienen en una federación, sus conceptos, operaciones y restricciones. A través de su descripción taxonómica se definen sus componentes y cómo se pueden resolver los conflictos que surgen al realizar una integración de bases de datos.

4.1 BASES DE DATOS RELACIONALES

Una base de datos relacional consiste de una colección de relaciones. Cada relación es una tabla en la que todas las entradas son datos atómicos como número, texto, fecha, etc. Las bases de datos relacionales están basadas en la teoría de conjuntos. Los elementos de una relación se denominan tuplas, el número de tuplas en una relación es la cardinalidad de la relación, y el número de atributos indica el grado de una relación.

4.1.1 Conceptos de Bases de Datos Relacionales

Las relaciones están conformadas por un conjunto de atributos que las definen. Son las tablas de la base de datos, así como también son tablas los distintos tipos de relaciones que se pueden generar mediante consultas a las relaciones base. En una base de datos relacional se encuentran los siguientes conceptos:

Esquema que define los atributos que conforman una relación, sus claves propias y ajenas. Es lo que corresponde al concepto de tabla. Interrelaciones**:** tipo de relación que poseen las relaciones entre sí y que definen el comportamiento de la base de datos. El tipo de relación está dado de acuerdo a la cardinalidad en las relaciones: uno-a-uno donde la clave primaria o propia de una relación se convierte en la clave ajena de la otra; relación uno-a-muchos, donde la clave de la relación del lado de uno se convierte en clave ajena de la relación del lado de muchos; y muchos-a-muchos, donde las claves propias de ambas relaciones generan una nueva relación.

El dominio D de una relación R es la colección de valores posibles para un determinado atributo. Un atributo posee un valor atómico.

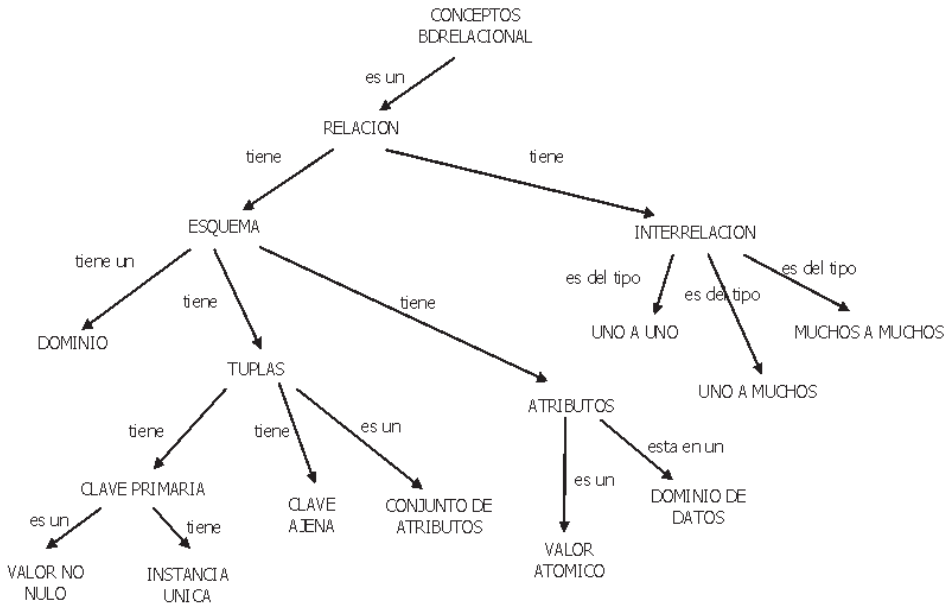
La clave primaria de la relación, que se define como un subconjunto C de los atributos de R, y cuyos valores no pueden ser repetidos, es mínima en el sentido de que en su composición no intervienen solamente los atributos estrictamente requeridos para identificar las tuplas de forma

Ùnicas.Una clave primaria puede ser simple (formada por un solo atributo) o compuesta (formada por más de uno) [8].

La **clave ajena** son aquellas claves primarias de una relación que se encuentran en otras relaciones, consiguiendo así la interrelación entre las relaciones que interesan.

A continuación, se muestra en la figura 6 el esquema ontológico que describe la taxonomía de los conceptos para las bases de datos relacionales.

Figura 6. Esquema Ontológico de los conceptos para Bases de Datos Relacionales



Los Axiomas de los conceptos para Bases de Datos Relacionales son:

Tabla 4. Axiomas de los conceptos para las Bases de datos Relacionales

Sentencia	LPO
Un concepto de base de datos relacional es una relación	∀ x ConceptoBDRelacional(x) => Tiene (x, Relación)
Una relación tiene esquemas y tiene interrelaciones	∀ x Relacion(x) => Tiene (x, Esquema) ∧ Tiene(x,Interrelaciones)
Un esquema tiene un dominio, tuplas y atributos	∀ x Esquema(x) => Tiene(x,Dominio) ∧ Tiene(x,Tuplas) ∧ Tiene(x,Atributos)
Una tupla tiene clave primaria, clave ajena y un conjunto de atributos	∀ x Tupla(x) => Tiene(x,Clave Primaria) ∧ Tiene(x,ClaveAjena) ∧ Tiene(x,ConjuntodeAtributos)
Un atributo es un valor atómico y está en un dominio	∀ x Atributo(x) => EsUn(x,ValorAtomico) ∧ EstáEn(x,Dominio)
Una interrelación es del tipo uno a uno o uno a muchos o muchos a muchos	∀ x Interrelacion(x) => EsdelTipo(x,UnaUno) ∨ EsdelTipo(x,UnoaMuchos) ∨ EsdelTipo(x,MuchosaMuchos)

4.1.2 Operaciones de Bases de Datos Relacionales

Las operaciones del modelo relacional se clasifican en consultas y actualizaciones. Las operaciones de actualización vienen dadas por insertar, modificar y eliminar una o más tuplas de una relación dada [19]. La operación Insertar proporciona una lista de valores de los atributos para una nueva tupla t que se insertará en una relación R. La operación Eliminar elimina una tupla t de una relación R. La operación Modificar sirve para cambiar los valores de uno o mas atributos en una tupla de una relación R. Surgen violaciones solo si se modifica un atributo clave, externa o primaria.

Las operaciones de consulta son las operaciones estándar del álgebra relacional, que permiten seleccionar tuplas y combinar tuplas a partir de varias relaciones, dando como resultado una nueva relación. Se clasifican en dos grupos: las operaciones de la teoría de conjuntos que son la Unión, la Interseccion, la Diferencia y el Producto Cartesiano, y las operaciones propias del álgebra relacional como seleccionar, proyectar y reunir [8]. En la figura 7 se muestra el esquema ontológico.

Figura 7. Esquema Ontológico de las Operaciones para Bases de Datos Relacionales

Los Axiomas de las Operaciones para bases de Datos Relacionales son:

Tabla 5. Axiomas de las Operaciones para las Bases de datos relacionales

Sentencias	LPO
Una operación de Base de Datos Relacional es una operación consultar relacional o una operación actualizar relacional	∀ x OperacionBDRelacional(x) => EsUna(x,OperacionConsultarR) V EsUna(x,OperaciónActualizarR)
Una operación consultar relacional puede ser un producto cartesiano o una diferencia relacional, o una unión de relaciones, o una división de relaciones o una selección de tuplas o una proyección de tuplas	∀ x OperaciónConsultarR(x) => PuedeSer(x,OperacionProductoCartesianoR) V PuedeSer (x,OperacionDiferenciadeRelaciones) V PuedeSer (x,OperacionUnionRelaciones) V PuedeSer (x,OperacionDivisiondeRelaciones) V PuedeSer (x,OperacionSelecciondeTuplas) V PuedeSer (x,OperacionProyeccionTuplas)
Una operación producto cartesiano genera un nuevo esquema	∀ x OperacionProductoCartesianoR(x) => Genera(x,NuevoEsquema)
Una operación diferencia de relaciones genera un nuevo esquema	∀ x OperacionDiferenciaRelaciones(x) => Genera(x,Nuevoesquema)
Una operación unión de relaciones genera un nuevo esquema	∀ x OperacionUniondeRelaciones(x) => Genera(x,Nuevoesquema)
Una operación unión de relaciones genera un nuevo esquema	∀ x OperacionUniondeRelaciones(x) => Genera(x,Nuevoesquema)
Una operación división de relaciones genera un nuevo esquema	∀ x OperacionDivisiondeRelaciones(x) => Genera(x,Nuevoesquema)
Una operación selección de tuplas genera una nuevo esquema	∀ x OperacionSelecciondeTuplas(x) => Genera(x,Nuevoesquema)
Una operación de proyección genera nuevo esquema	∀ x OperacionProyeccion(x) => Genera(x,NuevoEsquema)
Un nuevo esquema es un esquema de base de datos	∀ x NuevoEsquema(x) => EsUn(x,Esquema)
Una operación actualizar puede ser una operación insertar o una operación modificar o una operación eliminar	∀ x OperacionActualizar(x) => PuedeSer(x,OperacionInsertar) V PuedeSer (x,OperaciónModificar) V PuedeSer (x,OperacionEliminar)

4.1.3 Restricciones de las Bases de Datos Relacionales

A continuación se describen las restricciones a ser consideradas al realizar operaciones en las bases de datos, y en el caso de que se violen que acciones tomar. Particularmente, consideraremos las restricciones de integridad, llamadas también las reglas del negocio que gobiernan el universo del discurso en las bases de datos relacionales. Los tipos de restricciones de integridad son:

Restricciones de Dominio. Son las restricciones sobre los tipos de valores que puede tener un atributo. Especifican que el valor de cada atributo A debe ser un valor atómico del dominio dom(A) para ese atributo.

Restricciones de clave. El valor del atributo que identifica de manera única una tupla en una relación. Se especifica sobre relaciones individuales.

Restricciones de Integridad de Entidades. Establece que ningún valor de clave primaria puede ser nulo. Se especifica sobre relaciones individuales.

Restricción de Integridad Referencial. Se especifica entre dos relaciones, y permite mantener la consistencia entre las tuplas de las dos relaciones. Se lleva a cabo a través de las claves externas.

Restricciones generales o restricciones de integridad semántica, son aquellas que se establecen respecto a las solicitudes a realizar en las consultas o vistas, por ejemplo: el número máximo de horas que un profesor trabaja en un proyecto por semana es 40.

Las operaciones de actualización: insertar, modificar y eliminar no deben violar las restricciones de integridad especificadas anteriormente.

La operación de inserción puede violar cualquiera de los cuatro tipos de restricciones: de dominio, si proporciona un valor de atributo que no se encuentra en el dominio correspondiente; de clave, si el valor clave de la nueva tupla t ya existe en otra tupla de la relación r(R); de entidades si la clave primaria de la nueva tupla t es nula, y la integridad referencial si el valor de cualquier clave externa de t hace referencia a una tupla que no existe en la relación referida. Si una operación de inserción viola una o más restricciones se pueden realizar las siguientes acciones: rechazar la inserción o corregir la razón por la que se rechazó la inserción.

La operación Eliminar sólo puede violar la integridad referencial, si las claves externas de otras tuplas de la base de datos hacen referencia a la tupla que se va a eliminar. Existen tres acciones a tomar: 1) rechazar la eliminación; 2) tratar de propagar la eliminación, eliminando las tuplas que hacen referencia a la tupla que se desea eliminar; y por último, modificar los valores del atributo de referencia que provocan la violación, todos esos valores se pondrían nulos, siempre y cuando no sean claves, o se cambiarían a otra tupla válida.

En la operación Modificar surgen violaciones solo si se realizan

modificaciones a un atributo clave, externa o primaria, y se resuelven de la misma manera que para insertar o eliminar.

En la figura 8 se muestra el esquema ontológico que describe las restricciones de bases de datos relacionales.

Los Axiomas de las restricciones para las Bases de Datos Relacionales se describen en la tabla 6.

Figura 8. Esquema Ontológico de las Restricciones para Bases de Datos Relacionales

Tabla 6. Axiomas de las Restricciones para las Bases de datos Operacionales

Sentencia	LPO
La restricción de una BDRelacional puede ser una restricción de integridad referencial o una restricción de integridad semántica	∀ x RestriccionBDRelacional(x) => PuedeSer(x,RestricciónIntegridadReferencial) V PuedeSer(x,RestriccionIntegridadSemantica)
Una restricción de integridad semántica es un valor dado por el usuario	∀ x RestriccionIntegridadSemantica(x) => Esun(x,ValordeUsuario)
Una restricción de integridad referencial es una restricción de dominio	∀ x RestricciondeIntegridadReferencial(x) => EsUna(x,Restricciónde Dominio)
Una restricción de dominio determina los valores permitidos de los atributos	∀ x Restricciónde Dominio(x) => Determina(x,ValorPermitidodeAtributo)
La restricción de integridad de entidades determina un valor no nulo de clave	∀ x RestricciondeIntegridaddeEntidad(x) => Determina(x,ValornoNulodeClave)
La restricción de clave determina un valor único de la clave	∀ x RestricciondeClave(x) => Determina(x,ValorUnicodeClave)
La restricción de integridad referencial es una interrelación permitida	∀ x RestricciondeIntegridadReferencial(x) => EsUna(x,InterrelacionPermitida)

4.2 BASES DE DATOS ORIENTADAS A OBJETO

Las bases de datos orientadas a objetos están basadas en varios conceptos fundamentales: cada entidad del mundo real es modelada como un objeto. Cada objeto está asociado con un identificador único. Cada objeto tiene un conjunto de atributos y métodos; a su vez, el valor de un atributo puede ser un objeto o un conjunto de objetos. El conjunto de atributos de un objeto representa la estructura, y el conjunto de métodos el comportamiento. El estado de un objeto es accesado ó modificado a través del envío de mensajes al objeto para invocar los métodos correspondientes. Los objetos que comparten la misma estructura y comportamiento se agrupan en clases. Una clase representa una plantilla para un conjunto de objetos similares. Cada objeto es una instancia de una clase. Una clase puede ser definida como una especialización de una o más clases. Una clase definida como una especialización se le llama subclase y hereda atributos y métodos de sus superclases.

4.2.1 Conceptos de las Bases de Datos Orientadas a Objeto

Objeto: es cualquier cosa real ó abstracta acerca de la cual se almacenan datos y los métodos que controlan dichos datos. Un objeto está compuesto por el identificador de objetos, el constructor del tipo de objeto y el valor del objeto.

Identificador de Objetos: Es un concepto o valor que identifica al objeto de manera única y es inmutable. Se le genera al objeto en el momento de ser creado.

Tipo de Objeto: Es un conjunto de objetos que tienen un mismo comportamiento. Los tipos de objeto se definen a través de los constructores de tipo de objeto: átomos, tuplas, conjuntos, listas y arreglos.

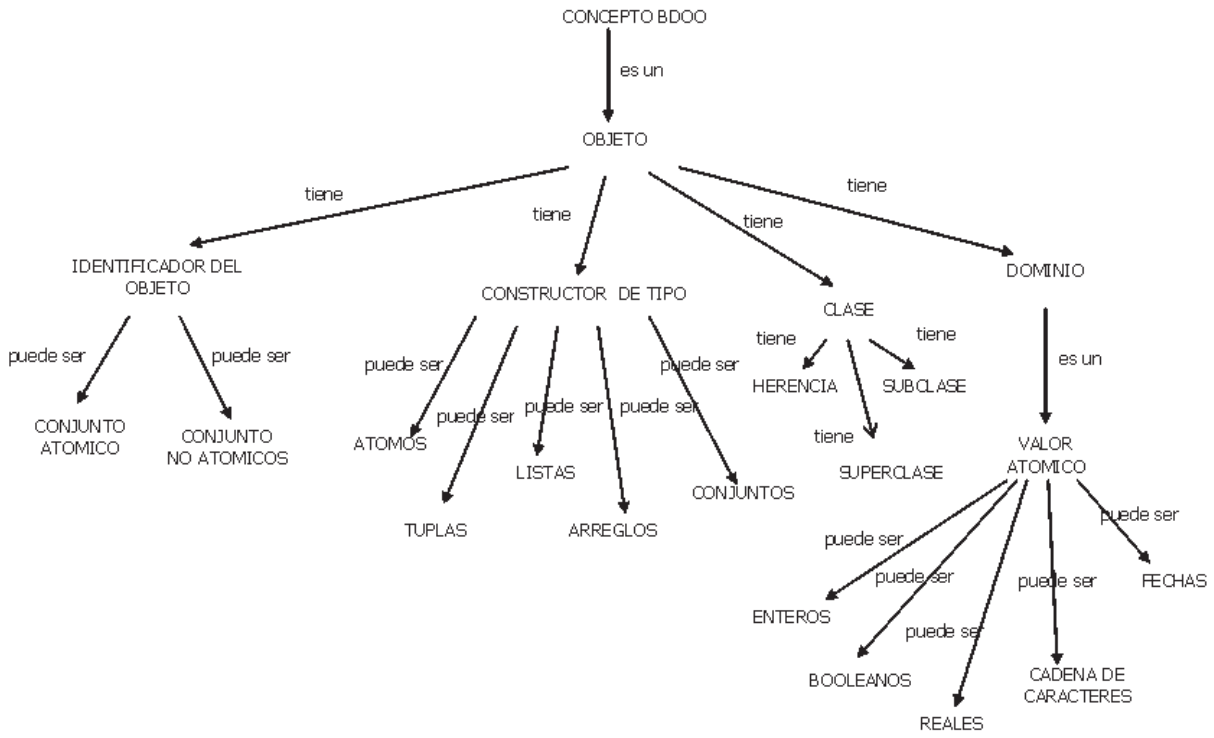
Los objetos tienen un **Dominio** que contienen los valores atómicos básicos disponibles como son: enteros, reales, cadena de caracteres, booleanos, fechas y cualquier otro tipo de datos que maneje la BDOO directamente. Además, puede tener valores no atómicos, que se refieren a un conjunto de objetos del mismo tipo como son tuplas, listas y arreglos.

Clase: Es la agrupación de objetos parecidos. Especifica una estructura de datos y los métodos operativos permitidos que se aplican a cada uno de sus objetos [17].

Jerarquía de clases: definición de clases a partir de clases predefinidos. Cuando las clases son parecidas se agrupan en subclases.

Herencia: tanto los atributos, como las operaciones de las clases y subclases se heredan.
En la figura 9 se muestra el esquema ontológico de los conceptos de las bases de datos orientadas a objeto.

Figura 9. Esquema Ontológico de los conceptos para las Bases de Datos Orientadas a Objeto



Los Axiomas de los conceptos para las Bases de Datos Orientadas a Objeto son:

Una Solicitud: invoca una operación específica, con uno ó más objetos como parámetros. En consecuencia, las implantaciones se refieren a los objetos como solicitudes.

Tabla 7. Axiomas de los Conceptos para las Bases de Datos Orientadas a Objeto

Sentencia	LPO
Un concepto de base de datos orientada a objetos es un objeto	∀ x ConceptoBDOO(x) => EsUn(x,Objeto)
Un objeto tiene identificador de objeto, constructor de tipo, clases y dominio	∀ x Objeto(x) => Tiene(x,IdentificadorObjeto) ∧ Tiene(x,ConstructordeTipo) ∧ Tiene(x,Clases) ∧ Tiene(x,Dominio)
Un identificador de objeto puede ser un conjunto atómico o un conjunto no atómico	∀ x IdentificadordeObjeto(x) => PuedeSer(x,ConjuntoAtómico) V PuedeSer (x,ConjuntoNoAtómico)
Un constructor de tipo puede ser un átomo o una tupla o una lista o un arreglo o un conjunto	∀ x ConstructordeTipo(x) => PuedeSer(x,Atomo) V PuedeSer (x,Tupla) V PuedeSer (x,Lista) V PuedeSer (x.Conjunto) V PuedeSer (x,Arreglo)
Un Dominio es un valor atómico	∀ x Dominio(x) => EsUn(x,ValorAtómico)
Un valor atómico puede ser un entero, booleano, real, cadena de caracteres, fecha	∀ x ValorAtomico(x) => PuedeSer(x,Entero) V PuedeSer (x,Booleano) V PuedeSer (x,Real) V PuedeSer (x,Cadena) V PuedeSer (x,Fecha)

4.2.2 Operaciones de las Bases de Datos Orientadas a Objeto

Los objetos son manipulados a través de los métodos que consisten de dos partes: un primer componente llamado signatura o interfaz de la operación, que especifica el nombre del método, los nombres y clases de los argumentos, y los resultados si existen; y un segundo componente, que especifica la implementación de la operación, que es un código escrito en un lenguaje de programación.

Las operaciones se invocan pasándole un mensaje a un objeto que incluye el nombre de la operación y los parámetros. Luego el objeto ejecutará el método para esa operación.

A continuación en la figura 10 se muestra el esquema ontológico de las operaciones de las bases de datos orientadas a objeto, antes descritas.

Los Axiomas de las operaciones para Bases de Datos Orientadas a Objeto se describen en la tabla 8.

4.2.3 Restricciones en Bases de Datos Orientadas a Objeto

Las bases de datos orientadas a objeto poseen las mismas restricciones que las bases de datos relacionales con la diferencia de que en lugar de referirse a tuplas y relaciones se refieren al objeto. Cada tipo de objeto tiene sus restricciones de integridad programadas en los métodos que crean, eliminan y actualizan los objetos.

Figura 10. Esquema Ontológico de las operaciones para Bases de Datos Orientadas a Objeto

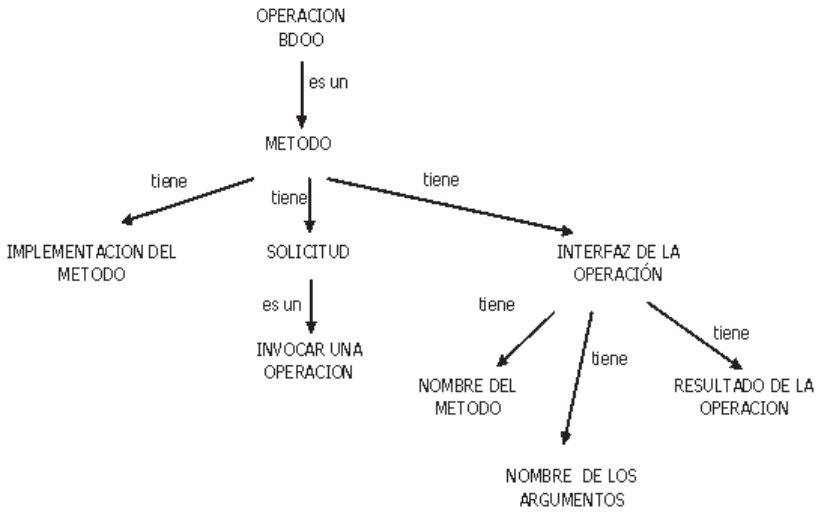


Tabla 8. Axiomas de las operaciones para Bases de Datos Orientadas a Objeto

Sentencia	LPO
Una operación de bases de datos orientadas a objeto es un método	∀ x OperacionBDOO(x) => EsUn (x,Metodo)
Un método tiene implementación, tiene solicitudes y tiene interfaz de la operación	∀ x Metodo(x) => Tiene(x,Implementacion) ∧ Tiene (x,Solicitud) ∧ Tiene(x,InterfazdeOperacion)
Una solicitud es una invocación de una operación	∀ x Solicitud(x) => EsUna(x,InvocaciónOperación)
Una interfaz de operación tiene nombre del método, tiene nombre de los argumentos y tiene resultados de la operación	∀ x InterfazdeOperación(x) => Tiene(x,NombredeMétodo) ∧ Tiene(x,NombredeArgumento) ∧ Tiene(x,ResultadosdeOperacion)

Las restricciones de Integridad Referencial de Objetos vienen dadas por: la restricción de dominio, que son las restricciones sobre los valores que pueden tomar las variables que conforman un objeto.

Restricción de integridad de objeto, establece que ningún valor del identificador de objeto puede ser nulo.

Asociaciones permitidas, especifican el tipo de asociaciones entre objetos. Las restricciones de integridad semánticas son aquellas que se establecen respecto a las solicitudes a realizar en las consultas. Las operaciones de actualización: insertar, modificar y eliminar no deben violar las restricciones de integridad especificadas.

A continuación en la figura 11 se muestra el esquema ontológico que describe las restricciones de las bases de datos orientadas a objeto.

Los Axiomas de las restricciones para Bases de Datos Orientadas a Objeto se describen en la tabla 9.

Figura 11. Esquema Ontológico de Restricciones para Bases de Datos Orientadas a Objetos

Tabla 9. Axiomas de las Restricciones para Bases de Datos Orientadas a Objeto

Sentencia	LPO
Una restricción de BDOO puede ser una restricción de integridad semántica o una restricción de integridad referencial	∀ x RestriccionBDOO(x) => PuedeSer(x,RestricciónIntegridadSemantica) V PuedeSer(x, RestricciónIntegridadReferencial)
Una restricción de integridad semántica es un valor de objeto dado por el usuario	∀ x RestricciónIntegridadSemantica(x) => EsUn(x,ValordefinidodeObjeto)
Una restricción de integridad referencial puede ser una restricción de dominio o una restricción de integridad de objeto o una asociación permitida	∀ x RestriccionIntegridadReferencial(x) => PuedeSer (x,RestricciondeDominio) V PuedeSer(x,RestriccionIntegridadObjeto) V PuedeSer (x,AsociaciónPermitida)
Una restricción de dominio es un valor permitido de objeto	∀ x RestricciondeDominio(x) => EsUn(x,ValorPermitido)
Una restricción de integridad de objeto es un valor no nulo	∀ x RestriccionIntegridadObjeto(x) => EsUn(x,ValornoNulo)
Una asociación permitida puede ser del tipo uno a uno o una a muchos o muchos a muchos	∀ x AsociaciónPermitida(x) => PuedeSer(x,UnoaUno) V PuedeSer (x,UnoaMuchos) V PuedeSer (x,MuchosaMuchos)

4.3 BASES DE DATOS DIFUSAS

La teoría de conjuntos difusos parte de la teoría clásica de conjuntos, añadiendo una función de pertenencia al conjunto, definida ésta como un número real entre 0 y 1. Dicha función de pertenencia μA (x), para cada conjunto difuso A, indica el grado en que la variable x está incluida en el concepto representado por dicho conjunto [21]. Así, un conjunto difuso A sobre un universo del discurso X está dado por:

A = {μA(x) /x : x ? X, μA(x) ? [0,1] ? }

Donde, μ es la función de pertenencia y μA(x) es el grado de pertenencia del elemento x al conjunto difuso A.

Por ejemplo, la variable difusa estatura_de_una_persona podría estar caracterizada por cada una de las etiquetas siguientes A = {bajo, medio, alto}, a las cuales se les pueden asociar una función de pertenencia μbajo(x), μmedio(x), μalto(x), respectivamente.
Por otro lado, la función de similitud establece que para cada dominio D, existe una relación de similitud que sirve para medir el parecido entre cada dos elementos del dominio.

Normalmente, los valores de similitud están normalizados en el intervalo [0,1], correspondiendo el 0 a "totalmente diferente" y el 1 a "totalmente parecido" o iguales [21]. Por tanto, una relación de similitud puede ser vista como una función sr, tal que:

sr : X×X -> [0,1]
sr(xi, xj) -> [0,1] tal que xi, xj ? X

Consideraremos en este trabajo que una Base de Datos Difusa es una Base de Datos Relacional que adicionalmente posee mecanismos para almacenar y procesar variables difusas. La forma de introducir información imprecisa es a través de: valores difusos en los atributos, grado de posibilidad y similitud en cada valor de un atributo, grado de posibilidad y similitud en toda la tupla de la relación, y grado de posibilidad y similitud en un conjunto de valores de diversos atributos.

4.3.1 Conceptos de Bases de Datos Difusas

Variables difusas: Las variables difusas pueden ser de 2 tipos:

Tipo 1: Estas son variables con "datos precisos", clásicos o crisp (tradicionales, sin imprecisión), que pueden tener etiquetas lingüísticas definidas sobre ellas. Las etiquetas lingüísticas pueden tener asociadas una función de pertenencia, posibilidad o de similitud.

Tipo 2: Son variables sobre "datos de dominio". En estos atributos se definen algunas etiquetas ("rubio", "pelirrojo", "castaño"...) y una relación de similitud definida entre ellas, de forma que esta relación indique en qué medida se parecen entre sí cada par de etiquetas.
Comparadores Difusos: Además de los comparadores comunes (=, <.>, etc.), existen los comparadores difusos que miden el grado de posibilidad, necesidad, pertenencia e incertidumbre [21].

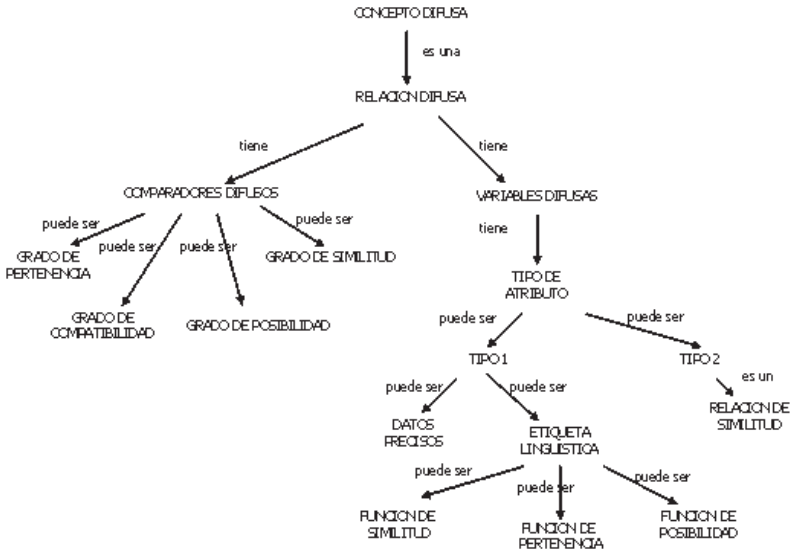
En la figura 12 se muestra el esquema ontológico de los conceptos de las bases de datos difusas.

Los Axiomas de los conceptos para Bases de Datos Difusas se describe en la tabla 10.

Tabla 10. Axiomas de los conceptos para Bases de Datos Difusas

Sentencia	LPO
Un concepto de Base de Datos Difusa es una relación difusa	∀ x ConceptoBDDifusa(x) => EsUna(x,RelacionDifusa)
Una relación difusa tiene comparadores difusos y variables difusas	∀ x RelacionDifusa(x) => Tiene(x,ComparadorDifuso) ∧ Tiene(x,VariableDifusa)
Un comparador difuso puede ser grado de pertenencia, o grado de compatibilidad o grado de posibilidad o grado de similitud	∀ x ComparadorDifuso(x) => PuedeSer(x,GradodePertenencia) ∨ PuedeSer(x,GradodeCompatibilidad) ∨ PuedeSer(x,GradodePosibilidad) ∨ PuedeSer(x,GradodeSimilitud)
Una variable difusa tiene tipo de atributo	∀ x VariableDifusa(x) => Tiene(x,TipodeAtributo)
Un tipo de atributo puede ser de tipo1 o de tipo2	∀ x TipodeAtributo(x) => PuedeSer(x,AtributoTipo1) ∨ PuedeSer(x,AtributoTipo2)
Un atributo tipo1 puede ser un dato preciso o una etiqueta lingüística	∀ x AtributoTipo1(x) => PuedeSer(x,DatoPreciso) ∨ PuedeSer(x,EtiquetaLinguistica)
Una etiqueta lingüística puede ser una función de posibilidad o función de pertenencia o función de similitud	∀ x EtiquetaLinguistica(x) => PuedeSer(x,FunciondePosibilidad) ∨ PuedeSer(x,FunciondePertenencia) ∨ PuedeSer(x,FunciondeSimilitud)
Un función de pertenencia indica el grado en que la variable x está incluida en el concepto representado por dicho conjunto y puede ser una función trapezoidal, Triangular, Gaussiana o Campana	∀ x FunciondePertenencia(x) => Mide(x,GradodePertenenciadelConcepto) ∧ [PuedeSer(x,FuncionTrapezoidal) ∨ PuedeSer (x,FunciónTriangular) ∨ PuedeSer(x,FuncionGaussiana) ∨ PuedeSer(x,FuncionCampana)]
Una función de similitud mide el parecido entre dos elementos	∀ x FunciondeSimilitud(x) => Mide(x,ParecidoentredosElementos)
Un atributo tipo2 es una relación de similitud	∀ x AtributoTipo2(x) => EsUn(x,RelaciondeSimilitud)
Una relación de similitud indica en qué medida se parecen entre sí un par de atributos	∀ x RelaciondeSimilitud(x) => Tiene(x,MedidadeSimilituddeAtributos)

Figura 12. Esquema Ontológico de los conceptos para bases de datos difusas



4.3.2 Operaciones de las Bases de Datos Difusas

En las bases de datos difusas se realizan las operaciones básicas del Álgebra Relacional: Unión, Diferencia, Producto Cartesiano, Proyección, División y Selección. Estas operaciones están definidas anteriormente para las bases de datos relacionales. Adicionalmente posee la función de pertenencia, que permite definir la variable difusa de acuerdo a su tipo de comportamiento.
La función de pertenencia viene dada por: Triangular, Función L, Función Gamma, Función Trapezoidal, Función S y Función Gaussiana.

En el producto cartesiano se encuentra el producto cartesiano de variables en un mismo dominio o en diferentes dominios. En el subyacen los mecanismos de razonamiento propios de la lógica difusa.

En la figura 13 se muestra el esquema ontológico de las operaciones de las bases de datos difusas.

Los Axiomas de las Operaciones para las Bases de Datos Difusas se describen en la tabla 11

Figura 13. Esquema Ontológico de las operaciones para Bases de Datos Difusas

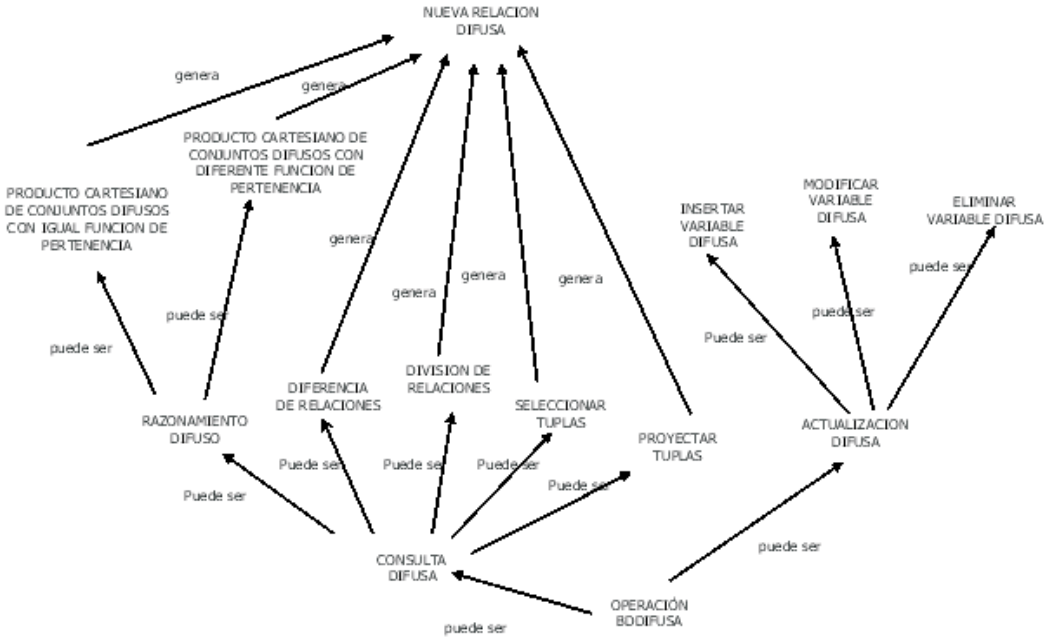


Tabla 10. Axiomas de los conceptos para Bases de Datos Difusas

Sentencia	LPO
Un concepto de Base de Datos Difusa es una relación difusa	∀ x ConceptoBDDifusa(x) => EsUna(x,RelacionDifusa)
Una relación difusa tiene comparadores difusos y variables difusas	∀ x RelacionDifusa(x) => Tiene(x,ComparadorDifuso) ∧ Tiene(x,VariableDifusa)
Un comparador difuso puede ser grado de pertenencia, o grado de compatibilidad o grado de posibilidad o grado de similitud	∀ x ComparadorDifuso(x) => PuedeSer(x,GradodePertenencia) ∨ PuedeSer(x,GradodeCompatibilidad) ∨ PuedeSer(x,GradodePosibilidad) ∨ PuedeSer(x,GradodeSimilitud)
Una variable difusa tiene tipo de atributo	∀ x VariableDifusa(x) => Tiene(x,TipodeAtributo)
Un tipo de atributo puede ser de tipo1 o de tipo2	∀ x TipodeAtributo(x) => PuedeSer(x,AtributoTipo1) ∨ PuedeSer(x,AtributoTipo2)
Un atributo tipo1 puede ser un dato preciso o una etiqueta lingüística	∀ x AtributoTipo1(x) => PuedeSer(x,DatoPreciso) ∨ PuedeSer(x,EtiquetaLinguistica)
Una etiqueta lingüística puede ser una función de posibilidad o función de pertenencia o función de similitud	∀ x EtiquetaLinguistica(x) => PuedeSer(x,FuncióndePosibilidad) ∨ PuedeSer(x,FuncióndePertenencia) ∨ PuedeSer(x,FuncióndeSimilitud)
Un función de pertenencia indica el grado en que la variable x está incluida en el concepto representado por dicho conjunto y puede ser una función trapezoidal, Triangular, Gaussiana o Campana	∀ x FuncióndePertenencia(x) => Mide(x,GradodePertenenciadelConcepto) ∧ [PuedeSer(x,FuncionTrapezoidal) ∨ PuedeSer (x,FunciónTriangular) ∨ PuedeSer(x,FuncionGaussiana) ∨ PuedeSer(x,FuncionCampana)]
Una función de similitud mide el parecido entre dos elementos	∀ x FuncióndeSimilitud(x) => Mide(x,ParecidoentredosElementos)
Un atributo tipo2 es una relación de similitud	∀ x AtributoTipo2(x) => EsUn(x,RelaciondeSimilitud)
Una relación de similitud indica en qué medida se parecen entre sí un par de atributos	∀ x RelaciondeSimilitud(x) => Tiene(x,MedidadeSimilituddeAtributos)

4.3.3 Restricciones de las Bases de Datos Difusas

Las restricciones son las mismas que para las bases de datos relacionales, y se le adicionan las restricciones propias de los conjuntos difusos: solapamiento entre variables y la pertenencia de variables al universo del discurso. Se muestran a continuación en la figura 14.

Los Axiomas de las restricciones para las Bases de Datos Difusas se describen en la tabla 11:

Figura 14. Esquema Ontológico de las restricciones para las Bases de Datos Difusas

Tabla 11. Axiomas de las restricciones para las Bases de Datos Difusas

Sentencia	LPO
Una restricción difusa tiene restricciones de integridad referencial y restricciones de pertenencia	∀ x RestriccionBDDifusa(x) => Tiene(x,RestriccionIntegridadReferencial) ∧ Tiene(x,RestriccionesdePertenencia)
Una restricción de integridad referencial puede ser una restricción de dominio difuso o una restricción de integridad de entidad difusa o una restricción de clave o una interrelación permitida	∀ x RestricciondeIntegridadReferencial(x) => PuedeSer(x,RestricionDominioDF) ∨ PuedeSer (x,RestricciondeIntegridadEntidadDF) ∨ PuedeSer (x,RestricciónClaveDF) ∨ PuedeSer (x,InterrelacionPermitida)
Una restricción de dominio es un valor permitido de variables difusas	∀ x RestriccionDominioDF(x) => EsUn(x,ValorPermitidoVariableDF)
Una restricción de integridad de entidades difusas es un valor no nulo de clave primaria	∀ x RestriccionIntegridadEntidadDF(x) => EsUn(x,ValorNoNuloCP)
Una restricción de clave es un valor único de tupla	∀ x RestriccionClaveDF(x) => EsUn(x,ValorUnicoTuplaDF)
Una interrelación permitida puede ser de uno a uno, o uno a muchos o muchos a muchos	∀ x InterrelacionPermitidaDF(x) => PuedeSer(x,UnoaUno) ∨ PuedeSer(x,UnoaMuchos) ∨ PuedeSer(x,MuchosaMuchos)
Una restricción de pertenencia puede ser un solapamiento entre variables o es el universo del discurso difuso	∀ x RestriccionesdePertenencia(x) => PuedeSer (x,SolapamientoentreVariablesDF) ∨ PuedeSer (x,UniversodelDiscursoDF)

4.4 BASES DE DATOS MULTIMEDIA

Una base de datos multimedia esta compuesta por objetos multimedia, texto, audio, video, imágenes, documentos, por lo tanto los sistemas de Bases de Datos Multimedia deben ser extensibles [16].

4.4.1 Conceptos para las Bases de Datos Multimedia

Las bases de datos multimedia requieren del uso de metadatos, los cuales describen los atributos de los objetos multimedia y les otorga significado, contexto y organización. A continuación hablaremos de algunos de ellos:
Los metadatos multimedia poseen características generales como: modelos en que se pueden representar, etiqueta del metadato, claves, notaciones y contenidos dependientes e independientes de metadatos.

Metadatos para texto: En ellos se incluye información sobre el texto, el cual puede ser utilizado para diseñar un libro, un periódico, o algo parecido. El metadato para texto incluye el tipo de texto, el número de páginas, el formato del texto y otra información como el número de capítulos y párrafos en cada capitulo, a su vez también puede incluir palabras claves.

Metadatos para imágenes: Pueden incluir datos para la descripción de las imágenes, por ejemplo: en una imagen ilustrando un océano con una palmera, una playa y una casa, el metadato para esta imagen podría describir textualmente la imagen X describe un océano con palmeras playas y casa. Esta información puede ser explotable a través de notaciones, por ejemplo cuando uno navega en la red las anotaciones acerca de las imágenes podrían ser activadas cuando se hace un clic sobre ellas. También existen técnicas de etiquetado de imágenes caracterizadas desde los metadatos para su posterior uso.

Figura 15. Esquema Ontológico de Conceptos para Bases de Datos Multimedia

Metadatos para Audio: El audio describe una información audible (clips) almacenada en una base de datos considerando dos partes, el de audio en sí y el texto que lo describe (es un resumen de lo que contiene). Por ejemplo, se puede almacenar en qué contexto se dio el audio.

Metadatos para Video: En este caso, estos datos son continuos, tanto este tipo de dato como el de audio se manejan de forma similar. Cuando el video es almacenado en la base de datos la información acerca del contenido y durabilidad del contenido se maneja en el metadato también. El metadato de video puede incluir imágenes del video.

En la figura 15 se mostró el esquema ontológico que describe los conceptos para las bases de datos multimedia.

Los Axiomas de los conceptos para las Bases de Datos Multimedia se describen en la tabla 12.

Tabla 12. Conceptos para las Bases de Datos Multimedia

Sentencia	LPO
Un concepto de base de datos multimedia puede ser un metadato para texto o un metadato para imagen o un metadato para audio o un metadato para video	∀ x ConceptoBDMultimedia(x) => PuedeSer(x,MetadatoTexto) V PuedeSer(x,MetadatoImagen) V PuedeSer(x,MetadatoAudio) V PuedeSer(x,MetadatoVideo)
Un metadato para texto tiene un texto y una descripción del texto	∀ x MetadatoTexto(x) => Tiene(x,Texto) ∧ Tiene(x,DescripciónTexto)
Un metadato para imagen tiene una imagen y una descripción de imagen	∀ x MetadatoImagen(x) => Tiene(x,Imagen) ∧ Tiene(x,DescripciónImagen)
Un metadato para video tiene un video y una descripción de video	∀ x MetadatoVideo(x) => Tiene (x,Imagen) ∧ Tiene (x,DescripciónImagen)
Un metadato para audio tiene audio y una descripción de audio	∀ x MetadatoAudio(x) => Tiene (x,Audio) ∧ Tiene (x,DescripciónAudio)

4.4.2 Operaciones de las Bases de Datos Multimedia

En estas bases de datos además de las operaciones propias de las bases de datos relacionales, poseen operaciones dadas por los tipos de datos que ellas manejan: video, imagen, texto y audio. Las operaciones de las bases de datos multimedia son: representación de datos, consultas y actualizaciones, búsqueda y edición, determinación de calidad de servicios procesados, y administración de seguridad/integridad.

Los datos multimedia, manejan el tiempo de la siguiente manera:

Instantes. Representan chronons simples, son utilizados para representar el instante en el que ocurre el hecho.

Periodos Temporales. Representa el conjunto de instantes contenidos entre dos instantes específicos.
Intervalos Temporales. Denota un tamaño de tiempo específico, sin precisar el instante de inicio y de fin. Por ejemplo, un día.
Conjunto de instantes. Es utilizado para representar un conjunto de instantes finitos, no necesariamente contiguos. Es muy utilizado para representar ocurrencias de tiempo que se repiten.
Elementos temporales. Es un conjunto finito de períodos temporales y representan una manera compacta de denotar un conjunto de espacios de tiempo no necesariamente contiguos.

En la figura 16 se muestran el esquema ontológico que describe las operaciones para las bases de datos multimedia.

Figura 16. Esquema Ontológico de Operaciones para Bases de Datos Multimedia

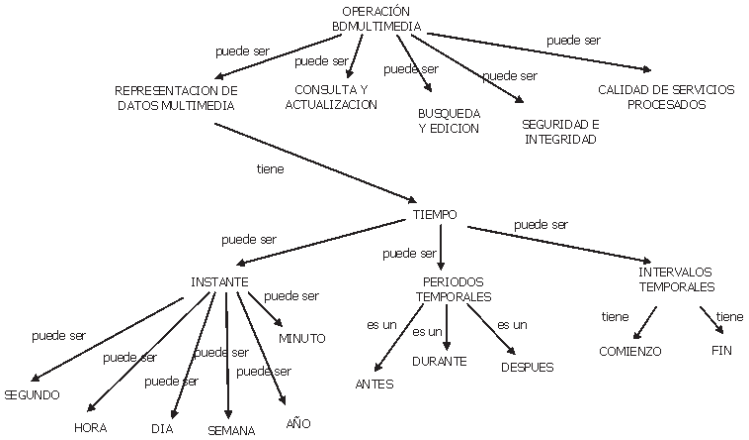


Tabla 13. Axiomas de las operaciones para las Bases de Datos Multimedia

Sentencia	LPO
Una operación de bases de datos multimedia puede ser una representación de datos multimedia o una consulta y actualización o una búsqueda y edición o seguridad e integridad o Calidad de servicios procesados	∀ x OperacionBDMultimedia(x) => Puedeser(x,RepresentacionDatosMM) V PuedeSer(x,ConsultaActualizacion) V PuedeSer(x,BusquedaEdicion) V PuedeSer(x,SeguridadIntegridad) V PuedeSer(x,CalidadServiciosProcesados)
Una representación de datos multimedia tiene tiempo	∀ x RepresentacionDatosMM(x) => Tiene(x,Tiempo)
El tiempo puede ser un instante un periodo temporal o un intervalo temporal	∀ x Tiempo(x) => PuedeSer(x,Instante) V PuedeSer (x,PeriodoTemporal) V PuedeSer(x,IntervaloTemporal)
Un instante puede ser un segundo o un minuto o una hora o un día o una semana o un año	∀ x Instante(x) => PuedeSer(x,Segundo) V PuedeSer (x,Minuto) V PuedeSer (x,Dia) V PuedeSer (x,Hora) V PuedeSer (x,Semana) V PuedeSer (x,Año)
El periodo temporal es un antes, un durante y un después	∀ x PeriodoTemporal(x) => EsUn(x,Antes) ∧ EsUn(x,Durante) ∧ EsUn(x,Después)
Un intervalo temporal tiene un comienzo y tiene un fin	∀ x IntervaloTemporal(x) => Tiene(x,Comienzo) ∧ Tiene (x,Fin)

Los Axiomas de las operaciones para las Bases de Datos Multimedia se describen en la tabla 13.

4.4.3 Restricciones de las Bases de Datos Multimedia

Las restricciones vienen dadas por la ocurrencia de eventos en un intervalo de tiempo específico, esto significa que se debe tomar en cuenta la sincronización. Para ello es fundamental considerar el momento en que los datos son presentados. Es decir, la sincronización tiene explícita la presentación de los datos en el momento adecuado y sitio adecuado.

Tiempo válido: tiempos en los que sucedió un determinado evento (o al intervalo durante el que persistió determinado estado). El tiempo válido de una proposición p es el conjunto de tiempos en los que se cree que p es verdadera.

Tiempo de transacción :conjunto de tiempos en los que p estuvo almacenada en la base de datos como verdadera.

En la figura 17 se muestra el esquema ontológico que describe las restricciones para las bases de datos multimedia.

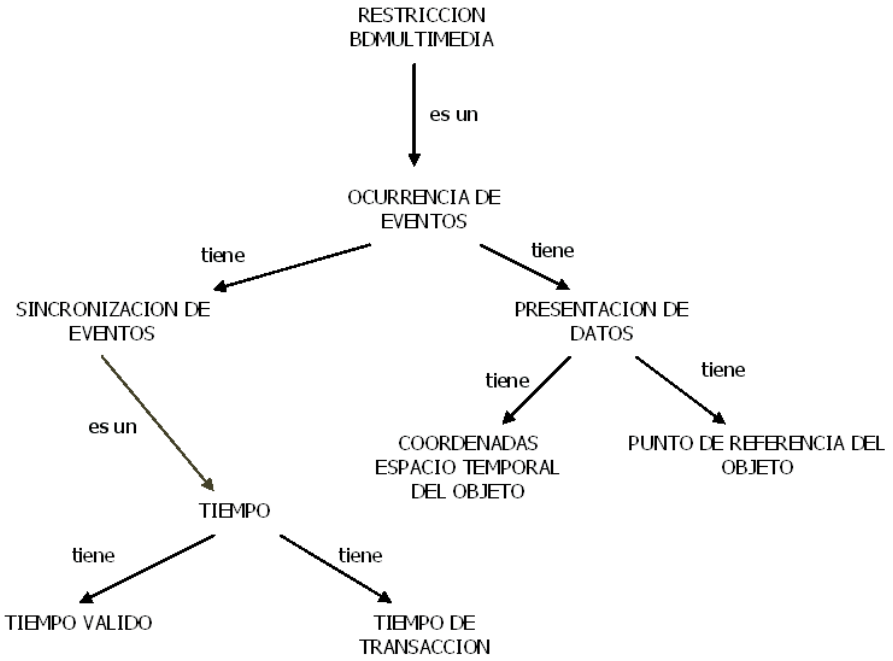
Los Axiomas de las restricciones para las Bases de Datos Multimedia se describen en la tabla 14.

4.5 BASES DE DATOS INTELIGENTES

En general, las bases de datos inteligentes están compuestas por una base de conocimientos y un mecanismo de razonamiento. La base de conocimientos está compuesta por reglas y hechos. Así, son los sistemas basados en reglas que mediante un esquema de razonamiento van determinando las reglas que se van activando hasta obtener la respuesta a una entrada dada (consulta, evento, etc.).

El esquema de razonamiento puede ser deductivo, inductivo o abductivo. Por ejemplo, se podría empezar con una evidencia inicial de una determinada situación para dar con una solución, o bien con hipótesis sobre posibles soluciones y volver hacia atrás para encontrar una evidencia que apoye la hipótesis, entre otras formas de razonamiento. Consideraremos dentro de las bases de datos inteligentes: las bases de datos activas, bases de datos deductivas y los sistemas basados en conocimiento [4].

Figura 17. Esquema ontológico de las restricciones para Bases de Datos Multimedia



4.5.1 Conceptos de las Bases de Datos Inteligentes

El proceso para realizar la inferencia y deducir conocimiento viene dado por dos conceptos:

Base de Conocimientos: Es una colección de hechos y reglas. Los hechos se pueden especificar de manera similar a como se especifican las relaciones en una base de datos relacional. Las reglas pueden ser referidas como "situación-acción" o "if-then". De esta forma, son invocadas por eventos o consultas que van apareciendo en la base de conocimiento (específicamente en la base de hechos). Las reglas se conectan una a otra por enlaces de asociación para formar redes de reglas.

Mecanismo de Razonamiento: Proceso de razonamiento a partir de los datos de entrada, considerando la base de conocimientos. Este mecanismo es genérico en el sentido de que puede aplicarse a diferentes dominios con solo cambiar la base de conocimientos.

La aplicación de las reglas cambian el estado del sistema, y por consiguiente, la base de conocimientos, habilitando unas reglas y deshabilitando otras. El intérprete de reglas usa una estrategia de control para encontrar las reglas disponibles y decidir qué regla debe aplicar basado en la forma de razonamiento utilizada (deductiva, inductiva, abductiva, etc.) [5].

A continuación, en la figura 18 se muestra su esquema ontológico.

Los Axiomas de los conceptos para las Bases de Datos Inteligentes se describen en la tabla 14.

4.5.2 Operaciones de las Bases de Datos Inteligentes

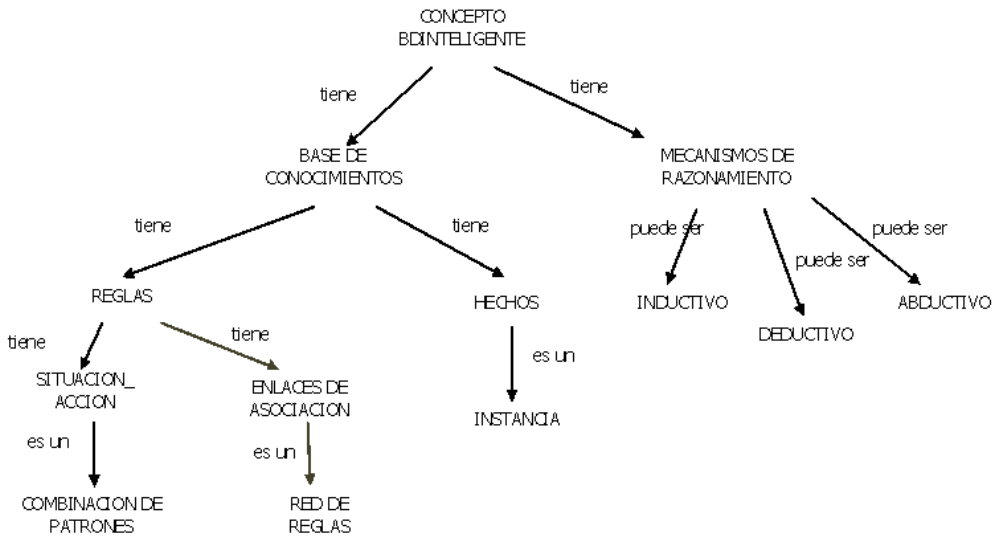
La máquina de razonamiento es quien controla el disparo de reglas. El ciclo se empieza con un evento o consulta y se para cuando no existen reglas aplicables. La máquina de razonamiento busca los elementos de la base de conocimientos que cumplen la condición. Luego aplica las reglas, ejecutando acciones, que involucran cambios en la base de conocimiento.

Existen diferentes estrategias de razonamiento: clásicamente

Tabla 14. Conceptos para las bases de datos inteligentes

Sentencia	LPO
Un concepto de bases de datos inteligentes tienen una base de conocimiento y un mecanismo de razonamiento	∀ x ConceptoBDInteligente(x) => Tiene(x,BasedeConocimiento) ∧ Tiene(x,MecanismodeRazonamiento)
La base de conocimientos tiene reglas y tiene hechos	∀ x BasedeConocimientos(x) => Tiene(x,Reglas) ∧ Tiene(x,Hechos)
Una regla tiene una situación-acción y tiene enlaces de asociación	∀ x Regla(x) => Tiene (x,Situación-Acción) ∧ Tiene(x,EnlaceAsociación)
Una condición es una combinación de patrones	∀ x Condicion(x) => EsUna(x,CombinaciondePatrones)
Un enlace de asociación es una red de reglas	∀ x EnlacedeAsociacion(x) => EsUna(x,ReddeReglas)
Un mecanismo de razonamiento puede ser deductivo, inductivo o abductivo	∀ x MecanismoRazonamiento(x) => PuedeSer(x,Deductivo) ∨ PuedeSer(x,Inductivo) ∨ PuedeSer(x,Abductivo)

Figura 18. Esquema Ontológico de Conceptos para Bases de Datos Inteligentes



pueden ser del tipo encadenamiento hacia adelante o encadenamiento hacia atrás. Encadenamiento hacia delante, parte de hechos para cumplir condiciones y ejecutar acciones (creando nuevos hechos). Encadenamiento hacia atrás parte de los estados deseados y trata de cumplir las condiciones necesarias para llegar a ellos [5].

En la figura 19 se muestra el esquema ontológico que describe las operaciones de las bases de datos inteligentes.

Los Axiomas para las operaciones de las Bases de Datos Inteligentes se describen en la tabla 15.

4.5.3 Restricciones de las Bases de Datos Inteligentes

Las restricciones en las bases de datos inteligentes vienen dadas por los tipos de conflictos en reglas, y son:

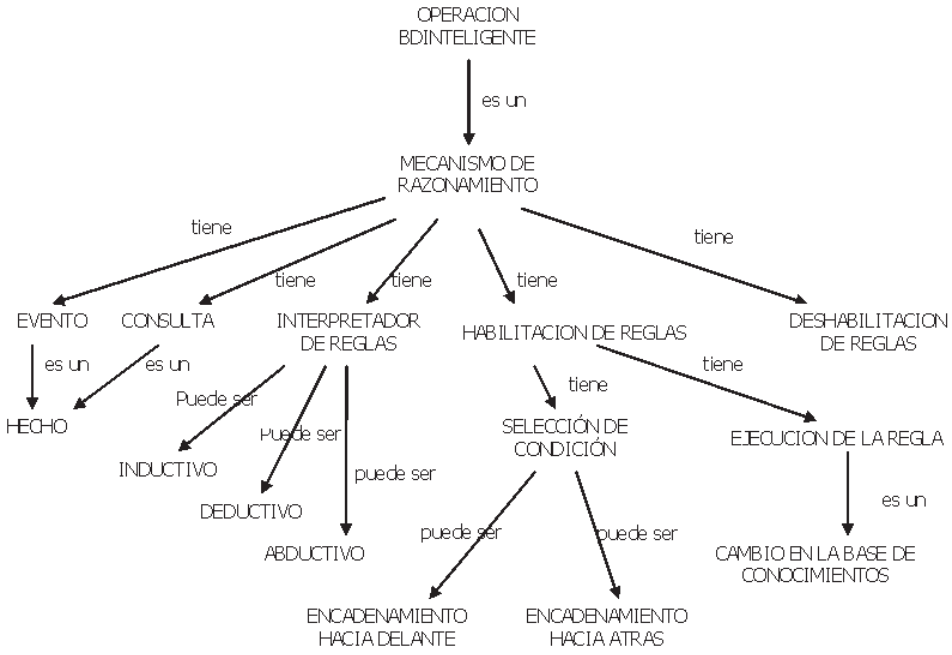
a. Disparo Simultáneo de Reglas, cuando un evento o consulta tiene diferentes reglas asociadas y el sistema permite que solo una regla se active. La solución puede ser: Selección aleatoria, uso de prioridades, (Ej.: tiempo de ejecución, selección concurrente de reglas), etc.

b. Contradicción entre reglas, cuando el evento o la consulta dispara una regla que es la negación del evento de la regla previamente disparada. En este caso se puede considerar inhibir esa activación.

Tabla 15.Axiomas para las operaciones de las Bases de Datos Inteligentes

Sentencia	LPO
Una operación de base de datos inteligentes es una mecanismo de razonamiento	∀ x OperacionBDIteligentes(x) => EsUn(x,MecanismodeRazonamiento)
El mecanismo de razonamiento tiene eventos y tiene consultas y tiene interpretador de reglas y tiene habilitación de reglas y deshabilitación de reglas	∀ x MecanismodeRazonamiento(x) => Tiene(x,Eventos) ∧ Tiene(x,Consultas) Tiene(x,InterpretadorReglas) ∧ Tiene(x,HabilitadorReglas) ∧ Tiene(x,DeshabilitadorReglas)
Un evento es un hecho	∀ x Evento(x) => EsUn(x,Hecho)
Una consulta es un hecho	∀ c EsUna(x,Consulta) => EsUn(x,Hecho)
Un interpretador de reglas puede realizar razonamiento deductivo, inductivo o abductivo	∀ x Interpretadorde Reglas(x) => PuedeSer(x,RazonamientoDeductivo) V PuedeSer(x,RazonamientoInductivo) V PuedeSer(x,RazonamientoAbductivo)
En el razonamiento deductivo se obtiene la conclusión de los hechos	∀ x RazonamientoDeductivo(x) => DeduceConclusion(x,Hechos)
El razonamiento inductivo de las conclusiones se obtienen los hechos	∀ x RazonamientoInductivo(x) => DeduceHechos(x,Conclusiones)
El razonamiento abductivo es cuando a partir de una hipótesis se obtienen conclusiones	∀ x RazonamientoAbductivo(x) => DeduceConclusion(x,Hipótesis)
La habilitación de reglas tiene selección de la condición y ejecución de la regla	∀ x HabilitadorReglas(x) => Tiene(x,SeleccióndeCondición) ∧ Tiene(x,EjecucióndeRegla)
La selección de la condición puede ser por encadenamiento hacia delante o encadenamiento hacia atrás	∀ x SeleccióndeCondicion(x) => PuedeSer(x,EncadenamientoHaciaDelante) V PuedeSer(x,EncadenamientoHaciaAtras)
La ejecución de la regla es un cambio en la base de conocimientos	∀ x EjecuciondeRegla(x) => EsUn(x,CambioenlaBC)

Figura 19. Esquema Ontológico de Operaciones en Bases de Datos Inteligentes



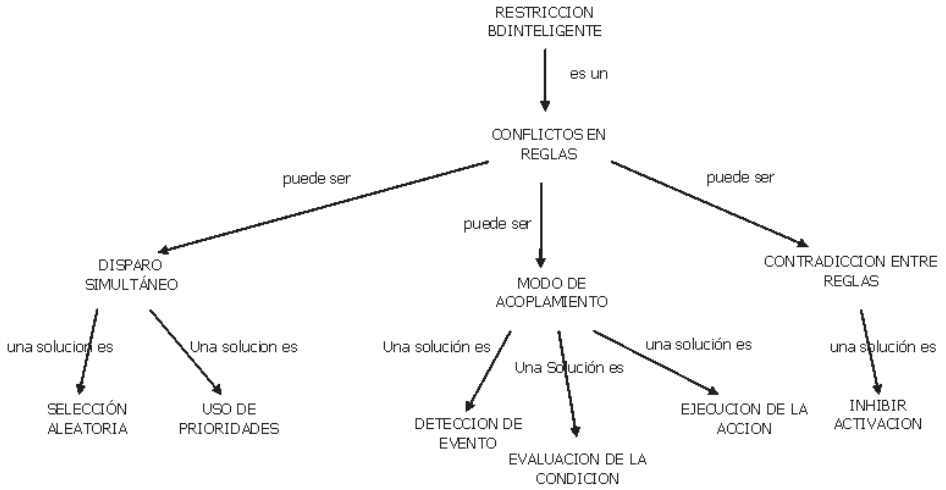
En la figura 20 se muestra el esquema ontológico.

Los Axiomas para las restricciones de las Bases de Datos Inteligentes se describen en la tabla 16.

5. CONCLUSIONES

En este trabajo se presentan los esquemas ontológicos que representan el proceso de integración de bases de datos, basados en la arquitectura de Shet&Larson [20] para bases de datos federadas, agregándole deducción y razonamiento a través de los esquemas ontológicos y sus axiomas. El desarrollo de las ontologías se utiliza como esquema que permita realizar la integración inteligente de una federación de bases de datos.
También hemos presentado los esquemas ontológicos de las bases de datos tradicionales como las relacionales, objeto y multimedia, y de las inteligentes como son los sistemas expertos, las bases de datos deductivas, las bases de datos activas y difusas.
Particularmente el modelo canónico debe poseer la habilidad de representación de los diferentes modelos de datos a nivel de sus estructuras, operaciones y restricciones de las bases de datos para poder conformar la federación, resolviendo los problemas de heterogeneidad que se puedan presentar.
Utilizamos la ontología como medio para representar el modelo canónico, ya que permite describir los conceptos en el dominio de las bases de datos y sus propiedades taxonómicamente, además de que con la ontología podremos diseñar mecanismos de manipulación sobre ella, se podrán desarrollar mecanismos de razonamiento y aprendizaje. Allí subyace la inteligencia y extensibilidad de nuestro Modelo de Integración Inteligente de Base de Datos Federadas.
En nuestra representación del Modelo de Integración Inteligente de Base de Datos Federadas encontramos inicialmente las taxonomías que describen los conceptos, operaciones y restricciones del proceso de integración de las bases de datos y luego se describen las taxonomías de cada una de las bases componentes de una federación a través de los conceptos, sus operaciones y sus restricciones.
Se presentan los axiomas que interpretan la taxonomía y permitirán llevar la ontología a un lenguaje de conocimientos. Con ellos se podrían realizar inferencias y extraer nuevos conocimientos.
Como trabajos futuros debemos traducir estos esquemas ontológicos a un lenguaje de Representación de

Figura 20. Esquema Ontológico de Restricciones en Bases de Datos Inteligentes



Sentencia	LPO
La restricción en las bases de datos inteligentes es un conflicto en reglas	∀ x RestricciónBDInteligente(x) => EsUn(x,ConflictoenReglas)
Un conflicto en regla puede ser un disparo simultáneo de reglas o una contradicción entre reglas	∀ x,y ConflictoenReglas(x,y) => PuedeSer(DisparoSimultaneo(x, y)) V PuedeSer(ContradiccionentreReglas (x, y)
Un disparo simultaneo de reglas se soluciona con la selección aleatoria de la regla o con el uso de prioridades	∀ x, y DisparoSimultaneodeReglas(x, y) => SuSolucionEs(Selección aleatoriadeRegla(x, y)) V SuSolucionEs(UsodePrioridades(x, y))
La contradicción entre reglas se soluciona con inhibir activación de la regla	∀ x, y ContradicciónentreReglas(x, y) => SuSolucionEs (x,InhibirActivaciondeRegla)

Conocimientos para construir la Base de Conocimientos del Modelo Canónico, y sobre él diseñar los mecanismos de manipulación del Modelo Canónico (razonamiento y aprendizaje). Además, podemos diseñar tareas de Minería de Datos sobre dicha Base de Conocimientos para extraer nuevos conocimientos derivados de la integración de las Bases de Datos, por ejemplo la creación de comunidades virtuales para las federaciones.

6. REFERENCIAS

[1] Abello A., Araque F., Samos J., Saltor F.; "Bases de Datos Federadas, Almacenes de Datos y Análisis Multidimensional", http://www.lsi.upc.es/~aabello/publications/home.html

[2] Abello A., M. Oliva, J. Samos, and F. Saltor; "Information System Architecture for secure Data Warehousing". In Proc. of the 3rd Int. Workshop on Engineering Federated Information Systems (EFIS), pag. 33-40. 2000

[3] Batini C., Lenzerini M.; "A comparative analysis of methodologies for database schema integration", ACM Computing Surveys 18, 4, December 1986.

[4] Bertino E., Catania B., Zarri Gian P.; "Intelligent Database System", Addison-Wesley. 2001.
http://ksi.cpsc.ucalgary.ca/KAW/KAW98/blazquez/

[5] Chapa V. Sergio; "Introducción a la Lógica Matemática", 1 9 9 9 http://delta.cs.cinvestav.mx/~schapa/red/intro_lm/logica1.html

[6] Condori Fernández, N; "FSQL. Fuzzy Estándar Query Lenguaje; SPCMagazine, Marzo 2002; http://www.spc.org.pe

[7] Corcho O., Fernandez-López M., Gomez-Perez A., "Methodologies, tools and languages for building ontologies. Where is their meeting point? Data & Knowledge Engineering 46 (2003) 41-64. Elsevier.

[8] Date, C. J.; "An Introduction to Database Systems", Volume 1, Fifth Edition. Reading, Mass: Addison-Wesley Publishing.1990

[9] Del Pino J., Roberto; "Bases de Datos Temporales". MABD 2003/2004

[10] Fernandez-Breis J., Martinez-Béjar R.; "A cooperative framework for integrating ontologies"; Elsiever Science Human Computer Studies 2002.

[11] Galindo Gómez J.; "Conjuntos y Sistemas Difusos (Lógica Difusa y Aplicaciones)". Departamento de Lenguajes y Ciencias de la Computación Universidad de Málaga. http://www.lcc.uma.es/~ppgg/FSS/FSSindex.pdf.

[12] Gruber, T. R. "A Translation Approach to Portable Ontology Specifications. KSL Report", 1993, http://ksl-web.stanford.edu/abstracts_by_author/Gruber,T..papers.html

[13] Weigand, H.; "Multilingual Ontology-Based Lexicon for News Filtering The TREVI Project", en K. Mahesh (1997): 138-159.

[14] Muñoz A., Aguilar J.; "Architecture for Distributed Intelligent Databases". IEEE, 13th Euromicro Conference on Parallel, Distributed and Network-based Processing, Euromicro-PDP 2005, pp 322-328

[15] Morales Eduardo, "Representación de Conocimiento". http://dns1.mor.itesm.mx/%7Eemorales/Cursos/RdeC/node197.html

[16] Patiño G, Angel M; "Bases de Datos Multimedia". Universidad de Castilla de la Mancha.

[17] Ruiz T., Martha Estela; "Desarrollo de SGBDOO en Oviedo3", www.Uniovi.es/~oviedo3/belen/jindbd96.html.6/Nov/1999

[18] Saltor F., Castellanos M, García-Solaco M; "Suitability of data models as canonical models for federated databases"; Universitat Politècnica de Catalunya.

[19] Silberschatz A., Korth H., Sudarshan S.; "Fundamentos de Bases de Datos". Mc Graw Hill, tercera edición. 1998

[20] Shet P, Larson J., "Federated Database System for managing distributed, heterogeneous and autonomous databases". ACM Computing Surveys 22, 1990 pp 183, 236

[21] Varas M., Urrutia A.; "Bases de Datos Difusas Modeladas con UML"; VIII Congreso Argentino de Ciencias de la Computación, 2002 pp 145, 155

[22] Varas M., Urrutia A.; "Bases de Datos Difusas Modeladas con UML"; VIII Congreso Argentino de Ciencias de la Computación, 2002 pp 145, 155

[23] Alvarez Carrión, G.; "Integración de esquemas en bases de datos heterogéneas fuertemente acopladas". Tesis Maestría, Universidad de las Américas, Puebla. 1999