Revista Venezolana de Computación
ISSN: 2244-7040
http://www.svc.net.ve/revecom
Vol. 3, No. 2, pp. 1-12, Diciembre 2016
Selección de los Mejores Artículos de CoNCISa 2016

Arquitectura para PostgreSQLf con Facilidades de Minería de Datos Descriptiva Difusa

Livia Borjas[1], Ana Aguilera[2], Rosseline Rodríguez[3], Francisca Losavio[4]
livacaro7@gmail.com, aaguilef@uc.edu.ve, crodrig@usb.ve, francislosavio@gmail.com

[1] Departamento de Informática, IUT FRP, Caracas, Venezuela
[2] Centro de Análisis, Modelado y Tratamiento de Datos CAMYTD, Universidad de Carabobo, Valencia, Venezuela
[3] Departamento de Computación, Universidad Simón Bolívar, Caracas, Venezuela
[4] Laboratorio de Modelos, Software y Tecnología (MoST), Universidad Central de Venezuela, Caracas, Venezuela

Resumen: En este trabajo se presenta una propuesta de arquitectura para PostgreSQLf, sistema gestor de bases de datos relacional difuso (SGBDRD) con capacidades de minería de datos descriptiva y representación difusa. Para la presente propuesta se aplicó el modelo de proceso de Diseño Arquitectónico Orientado a Metas, Aspectos y Calidad (DAOMAC), proceso que hace énfasis en la consideración explícita de los requisitos no funcionales desde el modelo de negocio y en diversos niveles de abstracción de todo el proceso de diseño arquitectónico hasta obtener una arquitectura inicial que garantice la trazabilidad entre componentes que representa funcionalidades y sus requisitos de calidad. Entre los resultados obtenidos, siguiendo DAOMAC, se destacan la descripción del modelo de negocio asociado al dominio del problema de implementación de sistemas gestores de bases de datos con requisitos funcionales de gestión de grandes volúmenes de datos y extracción de modelos con representación difusa. Además, se presenta la propuesta de una arquitectura general para PostgreSQLf, en su versión como arquitectura inicial orientada a metas, aspectos y calidad para esta familia de sistemas.

Palabras Clave: Arquitectura de Software; SGBDR; PostgreSQLf; Minería de Datos Difusa; Diseño Arquitectónico; DAOMAC.

Abstract: This work presents an architecture proposal for the fuzzy Relational Database Management System (RDBMS), PostgreSQLf, with capabilities of descriptive data mining and fuzzy representation. For the present proposal, the Architectural Design Process Oriented Design for Aspects and Quality (or Diseño Orientado a Metas, Aspectos y Calidad, DAOMAC) process model was applied; this is a process that emphasizes the explicit consideration of non-functional requirements from the business model; various levels of abstraction are considered during the whole architectural design process, until obtaining an initial architecture that guarantees the traceability of components representing functionalities and their quality requirements. Among the results obtained, according to DAOMAC, we highlight the description of the business model associated to the domain of the problem of implementation of database management systems with functional requirements for managing large volumes of data and extracting models with fuzzy representation. In addition, we present the proposal of a general architecture for PostgreSQLf, in its version as initial architecture, goals, aspects and quality-oriented, for this family of systems.

Keywords: Software Architecture; RDBMS; PostgreSQLf; Fuzzy Data Mining; Architectural Design; DAOMAC.

I. Introduction

Las Bases de Datos Relacionales, han facilitado la manipulación de datos en las organizaciones eficientemente. A pesar de sus grandes aportes en el manejo de la información, los Sistemas Gestores de Bases de Datos Relacionales (SGBDR) han presentado las siguientes situaciones: rigidez en la especificación y procesamiento de datos imperfectos, así como la acumulación de grandes volúmenes de datos, producto de la operación diaria de dichas bases de datos, sin ofrecer capacidades para su explotación rentable.

Por esta razón implementar SGBDR que provean capacidades de flexibilización en la especificación y manipulación de los datos, así como de mecanismos para la explotación de grandes volúmenes de datos, es un reto de investigación en la actualidad en la materia de gestión de datos [1][2][3].

Una manera de implementar este tipo de sistemas es integrando los SGBDR existentes con funcionalidades de gestión de datos difusos y de aplicación de las técnicas de Minería de Datos [4][5], que permiten extraer conocimiento implícito y novedoso con representación difusa. Existen tres (3)

arquitecturas de integración o acoplamiento [3] de estos SGBDR con estás funcionalidades especiales, algunas incorporadas a través de extensiones del lenguaje de consultas SQL con lógica difusa [1][6].

En los últimos veinte años [5], se han tenido diferentes experiencias en la implementación y aplicación de Sistemas Gestores de Bases de Datos Relacionales Difusos (SGBDRD), que usan SQLf y más recientemente con técnicas de minería de datos difusas utilizando diversos enfoques de integración [7]. Además, se han desarrollado diferentes aplicaciones que usan estas implementaciones [5].

Sin embargo, algunas de estas experiencias han usado procesos de desarrollo de software que no proveen en general mecanismos de conceptualización de requisitos funcionales (RF) difusos para sistemas o productos de minería de datos, ni de requisitos no funcionales (RNF) [8]. De igual manera no existe un marco arquitectónico de referencias que permita validar las características de calidad que estos productos presentan.

Por estas razones, en este trabajo se presenta una arquitectura inicial que sirva como marco de referencia para la implementación de estos SGBDR. Se espera que dicha arquitectura garantice la clara incorporación de requisitos funcionales y no funcionales en el proceso de construcción de un SGBDR Difuso (SGBDRD) desde el análisis del dominio, a fin de que provea mecanismos de trazabilidad entre sus modelos, vistas.

El presente trabajo se organiza de la siguiente forma: La Sección II expone los antecedentes de la investigación. La Sección III ofrece una breve explicación del modelo de proceso usado para el diseño arquitectónico. En la Sección IV se presenta la arquitectura inicial propuesta resultante de la aplicación de la metodología. Finalmente, en la Sección V se plantean las conclusiones y trabajos futuros.

II. ANTECEDENTES

Los antecedentes de la presente investigación incluyen: extensiones de SQL para incorporar gestión de datos difusos y capacidades de minería de datos; diferentes propuestas de arquitecturas según el tipo de acoplamiento usado, que permiten integrar las extensiones de SQL con los SGBDR; y finalmente, el proyecto PostgreSQLf, cuyo propósito es implementar una versión de PostgreSQL que permita la gestión de datos difusos.

A. SQLf

Se trata de una extensión de SQL que permite realizar consultas difusas sobre una base de datos tradicional. Las consultas difusas permiten especificar expresiones con lógica difusa en los lugares donde se usa la lógica clásica. La teoría de Conjuntos Difusos [9] es la base de la Lógica Difusa. En esta lógica, el valor de verdad de una condición está en el intervalo real [0,1]. El valor 0 se entiende como “completamente falso” y el valor 1 como “completamente cierto”. Los valores intermedios representan distintos grados de certidumbre o falsedad.

El uso de la lógica difusa permite gestionar datos de naturaleza vaga, imprecisa o incompleta [10], así como consultas difusas sobre datos tradicionales, los cuales están presentes en la lingüística humana y muchas veces expresan las preferencias de los usuarios [2][11]. Por ello han surgido varias extensiones del lenguaje de consultas SQL con lógica difusa, entre las cuales las más completas son: FSQL [12] y SQLf [1].

La presente investigación tiene especial interés en SQLf, por la variedad de consultas con condiciones flexibles, que permite especificar sobre Bases de Datos Relacionales. Los términos difusos de SQLf son incorporados en el sublenguaje de definición de datos DDL, del inglés Data Definition Language, y el sublenguaje de manipulación de datos DML, del inglés Data Manipulation Language incluyendo expresiones lingüísticas como: predicados, modificadores, comparadores, conectores y cuantificadores.

Por ejemplo, si se desea definir el término lingüístico “joven" sobre un dominio de edades comprendidas entre 0 y 120, se puede especificar un predicado difuso cuya función de membresía indique satisfacción completa (grado 1) para las edades menores que 30, para las edades en el intervalo comprendido de 30 a 60 años, un grado que decrece proporcionalmente (en línea recta) y para las edades mayores que 60 una completa insatisfacción (grado 0). Esta definición se expresa en SQLf mediante la instrucción CREATE FUZZY PREDICATE joven ON 0 .. 120 AS (0, 0, 30, 60). El gráfico de este tipo de definición, conocida como trapezoidal, puede observarse en la Figura 1.

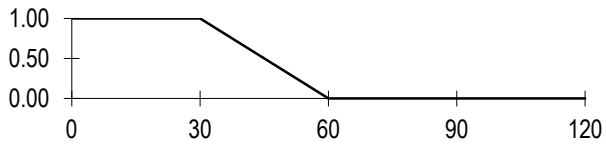

Figura 1: Definición Trapezoidal del Predicado Difuso *Joven*

La estructura básica de consulta de SQLf es el bloque multirelacional, cuya forma es SELECT <*attributes*> FROM <*relations*> WHERE <*fuzzy conditions*> WITH CALIBRATION [k|a|k,a]. Donde la cláusula WITH CALIBRATION es opcional, y permite la escogencia de las mejores respuestas. De esta forma, las condiciones (<*fuzzy conditions*>) de la cláusula WHERE pueden involucrar términos difusos definidos por el usuario, operadores y/o subconsultas difusas.

Entre sus características, SQLf permite la cuantificación lingüística sobre el conjunto de filas obtenidas en una consulta, a través del uso de estructuras de anidamiento y particionamiento con cuantificadores difusos. Así como también permite el uso de cuantificadores para calificar la cantidad de criterios de búsqueda satisfechos por filas.

Se han propuesto diversos mecanismos para la evaluación de las consultas en SQLf, destacándose el principio de derivación propuesto por Bosc y Pivert [13]. Este principio permite derivar consultas clásicas a partir de consultas difusas, de manera que el costo añadido por el procesamiento de un lenguaje difuso se mantiene lo más bajo posible.

En el presente trabajo se propone incorporar a SQLf capacidades para explotar los grandes volúmenes de datos presentes en almacenes históricos, producto de la operación de las bases de datos relacionales en diversas organizaciones. La

idea es que se puedan ejecutar consultas difusas sobre estos almacenes usando técnicas de minería de datos.

B. Extensiones de SQL para Minería de Datos

Algunas extensiones al lenguaje SQL se han realizado para soportar las tareas de minería de datos en el contexto de los SGBDR. Entre estas extensiones se destaca, con especial interés para la presente investigación, la propuesta de Timarán [3]. En ella se extiende SQL con primitivas de minería de datos en una arquitectura fuertemente acoplada sobre PostgreSQL. Los algoritmos de extracción de conocimiento forman parte del núcleo del SGBDR como operaciones primitivas, por lo que son ejecutados en el mismo espacio de direccionamiento que los datos, resultando en una extensión de SQL con muy buen desempeño en costo, y mostrando buen rendimiento y escalabilidad.

Otros esfuerzos se han realizado para incorporar reglas de asociación en lenguajes de consultas difusas a bases de datos tradicionales. Para el caso de SQLf, se incorporaron técnicas de análisis de datos a través del operador GROUP BY [14] para generar grupos difusos de una manera práctica y eficiente. Además, se han realizado desarrollos que comprueban la factibilidad de implementar estos operadores [15].

Otro grupo de extensiones [3] incluyen operadores unificados SQL-LIKE para soportar una tarea específica de minería de datos. También se proveen primitivas genéricas que ayudan al proceso de extracción de conocimiento sin soportar una tarea de minería de datos específica. Por otro lado, se han propuesto lenguajes de consulta para minería de datos con características similares al lenguaje SQL con la sintaxis SQL-LIKE. La debilidad de estas propuestas es que utilizan arquitecturas débiles y medianamente acopladas, produciendo bajo rendimiento y baja escalabilidad.

C. Arquitecturas de Acoplamiento

Para incorporar nuevas funcionalidades a un SGBDR se utilizan diferentes estrategias conocidas como arquitecturas de acoplamiento [5]. El tipo de arquitectura de acoplamiento usada en una extensión a un SGBDR es una decisión de diseño de mucha importancia, pues su selección podrá favorecer los RNF que definen la calidad de dicho producto. En la Figura 2 se pueden observar los tres tipos de arquitectura [3]: acoplamiento débil (*Loosely coupled*), acoplamiento medio (*Mildly coupled*) y acoplamiento fuerte (*Tightly coupled*). En trabajos previos se ha tenido diversas experiencias que muestran la factibilidad de implementar un SGBDRD con un desempeño adecuado [5].

Estas tres posibles arquitecturas de acoplamiento pueden ser usadas para integrar extensiones de SQL que incorporan el uso de requisitos funcionales considerados "novedosos", pues no son soportados por la mayoría de las herramientas conocidas, es decir la gestión de datos difusos y de minería de datos, que implican criterios no funcionales excluyentes entre sí, como son: rendimiento, escalabilidad y portabilidad [8].

Cada estrategia de integración tiene sus ventajas y desventajas [5][8]. En particular la alta portabilidad es la ventaja principal de los SGBDR débilmente acoplados, mientras en contraste el rendimiento es el mayor aporte del caso fuertemente acoplado, aunque se pierde en portabilidad. El acoplamiento medio ofrece la oportunidad de añadir a los SGBDR las características imprecisas de SQLf, sacrificando la migración. En la Tabla I, se resumen los RNF para estas tres arquitecturas que surgen ante la necesidad de gestionar requisitos funcionales "novedosos".

Figura 2: Arquitecturas de Acoplamiento

Tabla I: RNF para las Arquitecturas de Acoplamiento Derivados de RF "Novedosos"

Arquitectura de Acoplamiento	**Requisito No Funcional**	
	Ventaja	***Desventaja***
Débil	Portabilidad	Rendimiento y escalabilidad
Medio	Escalabilidad, Rendimiento	Portabilidad, mayor costo de desarrollo
Fuerte	Rendimiento y escalabilidad	sin portabilidad, alto costo de desarrollo

Para el presente trabajo se está interesado de manera especial en la arquitectura de acoplamiento fuerte.

D. Proyecto PostgreSQLf

Un antecedente directo a la presente investigación es el Proyecto PostgreSQLf, desarrollado en la Universidad de Carabobo, desde el año 2006, el cual agrega capacidades para expresar consultas flexibles a bases de datos al SGBD Objeto-Relacional, PostgreSQL, usando el lenguaje SQLf [15].

Entre las capacidades del lenguaje SQLf que se han incorporado están: predicados difusos, operadores difusos AND y OR, modificadores difusos, consultas particionadas difusas en las cláusulas GROUP BY y HAVING, vistas y operación de unión difusas, creación de cuantificadores difusos y consultas cuantificadas difusas de tipo bloque simple, comparadores difusos, subconsultas difusas en la cláusula FROM, bloques anidados EXISTS-LIKE y ANY-LIKE [15].

En [16] se realiza una extensión del motor de consultas del PostgreSQLf usando una arquitectura de acoplamiento fuerte pero no propone un diseño arquitectónico orientado a metas que considere los aspectos y la calidad. Tampoco se incluyeron operadores para el análisis de agrupamiento difuso ni la especificación de reglas de asociación difusas.

En el marco de este proyecto, se ha propuesto desarrollar una extensión de SQLf para realizar análisis descriptivo difuso desde PostgreSQLf, con una arquitectura fuertemente acoplada

[8]. Se pretende añadir al SGBDRD operadores para extraer Reglas de Asociación Difusas y Grupos Difusos. La idea es proveer primitivas para realizar la tarea del análisis de agrupamiento, así como, especificar reglas de asociación difusas, dentro de la instrucción SELECT.

En este trabajo se presenta una propuesta arquitectónica para el SGBDRD PostgreSQLf, en el cual se incorpora la extensión de SQLf con las nuevas primitivas que soportan análisis de asociación y de agrupamiento difuso, de manera que se aprovechen los beneficios de rendimiento (eficiencia) y escalabilidad propias de la arquitectura de acoplamiento fuerte.

III. Principios de Diseño

Una Arquitectura de Software [17][18] es el conjunto de componentes, conectores y relaciones que representan la estructura, comportamiento y propiedades esenciales de un sistema de software, de manera global, con un alto nivel de abstracción. Los componentes son los elementos donde se llevan a cabo los cómputos, las partes representan los roles, los puertos describen los puntos de interacción entre el entorno y las partes internas; y los conectores constituyen los medios por los cuales interactúan los componentes y sus partes. En conjunto, estos elementos conforman la estructura estática o lógica de un sistema de software, de tal manera que la arquitectura describe la organización del sistema de software, las relaciones con otros componentes, el ambiente y los principios que gobiernan su diseño y evolución [19].

El diseño de una arquitectura de software [17] es un tema de interés en la actualidad; contemplan aspectos de calidad que influye en la capacidad evolutiva de la arquitectura o base que soporta el sistema. Son varios los trabajos que tratan el problema de diseñar una arquitectura del software [20][21]. En particular existen esfuerzos [22] que se enfocan en el análisis de RNF para el diseño arquitectónico, por ser estos los que orientan dicho diseño.

En el presente trabajo se usa como punto de partida el modelo de proceso para Diseño Arquitectónico Orientado a Metas, Aspectos y Calidad (DAOMAC) [23]. Este proceso es una propuesta para el diseño de una arquitectura inicial, centrado en la especificación de los requisitos de calidad de producto de software, asociados a los RNF relativos al dominio, especificados según el estándar ISO/IEC 25010 [24]. DAOMAC garantiza la correspondencia o trazabilidad entre el modelo de negocio y el modelo de la arquitectura inicial del sistema de software, conforme a los RNF que intervienen y que dirigen la construcción de la misma. En este modelo es esencial considerar los aspectos no funcionales desde tempranos niveles de abstracción, de acuerdo al enfoque de orientación a aspectos [22], como lo es el modelo de negocio. La orientación a aspectos es un paradigma de la Ingeniería de Software que sugiere tratar "temprano" en el ciclo de desarrollo todas aquellas incumbencias que son aspectos del sistema no percibidos directamente por el usuario, como es el caso de los RNF. En DAOMAC se hace uso de este principio para garantizar la trazabilidad de los RNF entre los modelos construidos, en cada nivel de abstracción; el uso de una especificación estándar [24] de las propiedades de calidad relativas a los RNF facilita la comunicación entre los miembros del equipo de desarrollo.

Para comprender mejor los planteamientos del proceso DAOMAC, se presentará de manera general algunos conceptos relacionados. En primer lugar, se define como *Arquitectura Inicial* (*AI*), la estructura computacional que posee los componentes principales del sistema de software, a partir de la cual se articulará el resto del sistema [17][19]. Una consideración fundamental para diseñar y construir una AI es tomar en cuenta como componentes específicos de la AI las propiedades funcionales conocidas como RF y los RNF que debe cumplir el sistema desde tempranas etapas del ciclo de vida del software.

En segundo lugar, dentro de una organización se puede definir un proceso [25] como una colección de actividades, que toma una o varias clases de entradas y genera una salida de valor para el cliente. Por tanto, un *Proceso de Negocio* [26] es el conjunto vinculado y natural de actividades basadas en habilidades y competencias de trabajo, que incluyen interacciones entre actores, recursos utilizados y vínculos de dependencias entre estos. Dichas actividades son ejecutadas de acuerdo a reglas preestablecidas que permiten satisfacer ciertas metas [27].

Dado que en una organización las operaciones suelen ser automatizadas, resulta útil concebir el negocio como soporte al proceso de Ingeniería de Software [22]. Por tal razón, el *Modelo de Negocio* (*MN*) es considerado como una vista abstracta y simplificada, en términos de metas y factores de importancia, que refleja la lógica de la compleja realidad del funcionamiento del negocio de una organización o entidad [23]. El MN es un punto inicial para la elaboración del sistema de software de mucha utilidad para la especificación de los requisitos globales que el sistema debe satisfacer. Así, el *Modelo de Proceso de Negocio* (*MPN*) [28] se define como la representación gráfica de los procesos organizacionales, o red de actividades y controles de flujos, que describe su funcionamiento.

Una AI [22] es el modelo en el cual se identifican los componentes orientados a aspectos, derivados de RNF y los componentes funcionales del sistema, derivados de RF. En la Figura 3, se observa que el modelo de proceso DAOMAC, está conformado por un "paquete" con cuatro disciplinas, cada una de las cuales es descrita en otro paquete. Las cuatro disciplinas son: Construcción del Modelo de Negocios, Obtención del Modelo de Metas, Determinación de las Metas No Funcionales Transversales y Construcción Orientada a Aspecto de la Arquitectura Inicial. Estas disciplinas se interrelacionan entre sí, de acuerdo a un modelo iterativo y los modelos obtenidos en cada nivel de abstracción son la entrada y punto de partida para construir el modelo del nivel de abstracción siguiente.

Estos modelos son representados usando diferentes lenguajes de notación gráfica: MN debe ser expresado en el lenguaje BPMN, (*Business Process Model and Notation*) [30][31], el Modelo de Metas (MM) en el Lenguaje de Requisitos Orientado a Metas o GRL (*Goal-oriented Requirements Language)* [32], y AI en la notación estándar SPEM (*Software Process Engineering Metamodel*) [33] basada en el Lenguaje Unificado de Modelado, UML (*Unified Modelling Language*) [34].

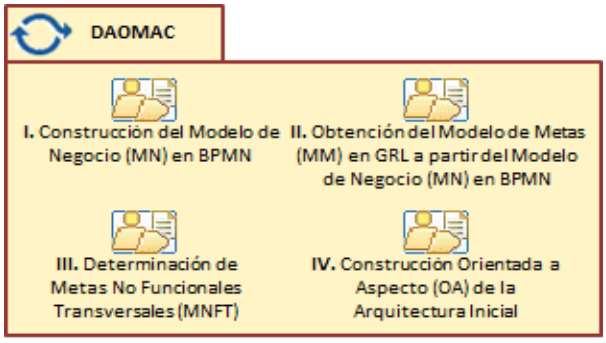

Figura 3: Proceso DAOMAC [29]

MM [23] incluye el conjunto de metas no funcionales transversales y su refinamiento en el diagrama o grafo jerárquico de interdependencias, ambos expresados como un *Softgoals Interdependency Graph* (SIG) [35][36]. Estas interdependencias son identificadas como operacionalizaciones concretas que permiten derivar componentes arquitectónicos; una *operacionalización* corresponde a un mecanismo arquitectónico específico que satisface o es una solución para el cumplimiento del *softgoal* (o meta no funcional). Estás metas están directamente relacionadas con los requisitos de calidad del sistema. El MM se basa en la Ingeniería de Requisitos Orientada a Metas [37]. MM se utiliza como enlace entre el modelo de negocio expresado en BPMN y el modelo de arquitectura inicial del sistema de software.

Esto se observa detalladamente en la Figura 4, la cual describe que el Modelo de Calidad del Producto, construido a partir del estándar ISO/EC 25010, el cual interviene en todos los niveles de abstracción del proceso DAOMAC.

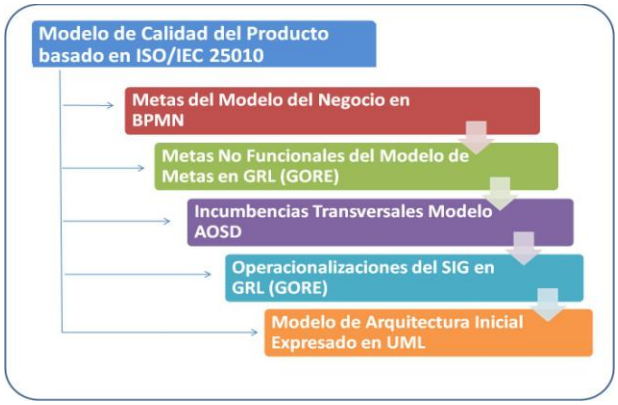

Figura 4: Niveles de Abstracción en el Proceso de DAOMAC

En DAOMAC, el MN es el punto de partida para construir la arquitectura inicial. Luego de aplicar varias transformaciones sobre el MN, se construye el modelo de metas, para finalmente obtener un modelo de arquitectura inicial. Este proceso presenta la siguiente secuencia transformacional:

1) En el MN, se identifican Tareas Funcionales y No Funcionales y los vínculos entre ellas. Las Tareas No Funcionales (TNF) pueden dar origen a Metas No Funcionales (MNF), denotadas por el estereotipo "MNF" del Lenguaje BPMN. Las MNF a nivel de BPMN indican objetivos de calidad asociados a una tarea [30][31].

2) Luego se construye el modelo de metas expresado en GRL, para hacer la transición entre las "MNF" del modelo de negocio y los requisitos no funcionales o *softgoals* del sistema y así identificar las Metas No Funcionales Transversales (MNFT), y su refinamiento en el SIG como operacionalizaciones concretas [23]. Estas MNFT del MM corresponden de manera directa con las Incumbencias Transversales (*Softgoals*) del Diagrama de Interdependencias de *Softgoal* o Modelo AOSD (*Aspect-Oriented Software Development*) [23]. Muy importante resaltar que la MNFT identificada como principal es el requisito no funcional transversal considerado de mayor prioridad para dicho sistema.

3) Seguidamente, las Operacionalizaciones del SIG expresadas en GRL, corresponden, en parte, con mecanismos o soluciones arquitectónicas [23], descritas y evaluadas de acuerdo a sus atributos o propiedades de calidad, para la selección de una opción adecuada entre varias alternativas. Finalmente, estos mecanismos/soluciones arquitectónicas se convertirán en "Componentes Aspectos", es decir, componentes orientados o tratados como aspectos (estereotipados en UML como «Aspect»), y en los componentes funcionales, obteniendo así el Modelo Arquitectónico Inicial expresado en UML como vista lógica de componentes y conectores [38].

A continuación, se detallan estas tres fases transformacionales del proceso DAOMAC.

B. Construcción del Modelo de Negocio en BPMN

El modelo de negocio expresado en BPMN identifica los vínculos entre tareas funcionales y no funcionales [39]. Para ello se realizan los siguientes pasos:

1) Crear los *Pools* (actores) y *Lanes* (roles de los actores) del Modelo de Negocio.
2) Definir las Tareas Funcionales y Subprocesos en los *Pools* y *Lanes*.
3) Incorporar Compuertas Exclusivas entre las diferentes Tareas Funcionales y Subprocesos que subyacen en los *Pools* y *Lanes*.
4) Identificar Tareas No Funcionales y asociarlas a Tareas Funcionales y subprocesos mediante el estereotipo «MNF».
5) Establecer los Flujos de Secuencia y de Mensajes.

C. Obtención del Modelo de Metas en GRL

En el MM expresado en GRL se identifican las metas no funcionales transversales y su refinamiento en el SIG como operacionalizaciones específicas [23]. Se construye después de hacer la transición entre las MNF del modelo de negocio y los requisitos no funcionales del sistema de software. Para construir el MM se realizan los siguientes pasos:

1) En base al MN, definir la Meta Funcional Medular o Principal asignando prioridades y vínculos de dependencias del MM.
2) En base a los *Pools* y *Lanes* del MN, definir los Actores y Roles en el MM.
3) Identificar *Pools*, *Lanes*, Subprocesos, Tareas y Asociaciones, Objetos de datos y Compuertas en BPMN y

hacerlas corresponder con Metas Funcionales, Metas No Funcionales y Tareas en GRL.

4) En base a los elementos de los *Pools* y *Lanes* en BPMN, establecer vínculos en GRL

D. Determinación de las Metas no Funcionales Transversales

Las MNFT son asociadas a mecanismos/soluciones arquitectónicas, las cuales son descritas de acuerdo a sus atributos o propiedades de calidad cumpliendo con el estándar ISO/EC 25010 [24] y evaluadas para la selección de una opción adecuada entre varias alternativas. Los pasos de esta fase son:

1) En base al MN, definir la Meta Funcional Medular o Principal asignando prioridades y vínculos de dependencias del MM.

2) En base a los *Pools* y *Lanes* del MN, definir los Actores y Roles en el MM.

3) Identificar *Pools*, *Lanes*, Subprocesos, Tareas y Asociaciones, Objetos de datos y Compuertas en BPMN y hacerlas corresponder con Metas Funcionales, Metas No Funcionales y Tareas en GRL.

4) En base a los elementos de los *Pools* y *Lanes* en BPMN, establecer vínculos en GRL.

E. Construcción Orientada a Aspectos (OA) de la Arquitectura Inicial

Para construir la arquitectura inicial se realizan los pasos:

1) Elegir la operacionalización de mayor relevancia desde el punto de vista arquitectónico, para cada MNFT, en orden de prioridad.

2) Describir los mecanismos/soluciones arquitectónicas asociados a la operacionalización seleccionada.

3) Comparar los mecanismos/soluciones arquitectónicas descritas considerando el modelo de calidad ISO/IEC 25010 [24], adaptado al dominio.

IV. ARQUITECTURA DE POSTGRESQL

PostgreSQL fue inicialmente desarrollado en el Departamento de Ciencias de la Computación de la Universidad de California, Berkeley [40]. A partir de 1996 deja de ser un proyecto académico de dicha universidad y pasa a manos de una comunidad desarrolladores de todo el mundo, quienes dotaron al sistema de una alta consistencia, uniformidad y seguridad. El grupo de desarrollo actual de este sistema, se conoce como *PostgreSQL Global Development Group*. Actualmente, PostgreSQL es un servidor de base de datos de código abierto muy avanzado cuya licencia permite ser usado, modificado y distribuido sin costo alguno.

En esta sección se describirá la arquitectura del SGBD Objeto-Relacional PostgreSQL, a fin de facilitar la presentación de la propuesta de extensión.

A. Estructura Interna de PostgreSQL

Una propuesta para modelar la arquitectura de PostgreSQL fue realizada en [32], la cual la presenta como una arquitectura Cliente-Servidor, que está dividida en tres subsistemas: el cliente (Client), el servidor (Server) y el gestor de almacenamiento (Storage Manager). En la Figura 5 se muestra la organización de tales subsistemas.

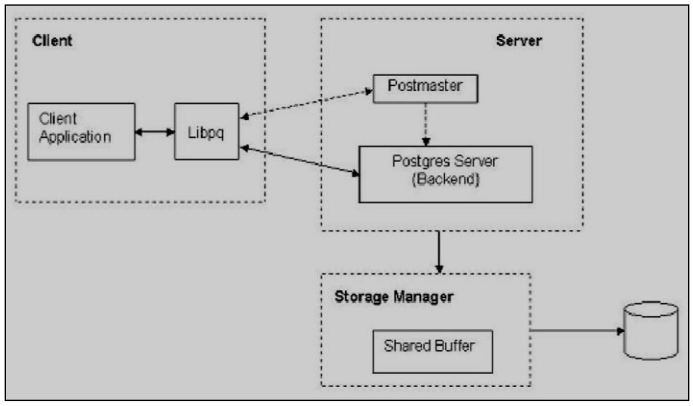

Figura 5: Estructura General de PostgreSQL [40]

El cliente consiste de cualquier aplicación que haga uso del SGBD, estableciendo una conexión al mismo por medio del componente Libpq. Este último es una interfaz en C nativo que gestiona la comunicación entre el SGBD y la aplicación cliente, enviando las consultas de este último al servidor. En la Figura 4 se muestra el funcionamiento de este subsistema.

El servidor lo integran dos componentes: el *Postmaster* y el *Postgres Server* o *backend*. El *Postmaster* está encargado de aceptar cada una de las peticiones del cliente que desee establecer una conexión, realiza las tareas de autenticación y control de acceso, y establece la conexión entre el cliente y un nuevo proceso o hilo del *Postgres Server*. De esta forma, cada cliente que haga uso del SGBD estará conectado a exactamente un proceso del servidor. Por otro lado, el *Postgres Server* maneja las consultas y tareas establecidas por el cliente.

El gestor de almacenamiento es responsable de toda la gestión y control de los recursos de almacenamiento en el SGBD, incluyendo *buffers* de memoria compartidos, así como, gestión de archivos, control de consistencia y manejo de *locks* sobre los archivos.

Dado que el servidor es quien representa el motor del SGBD, posteriormente, en [41] se estableció la estructura interna del *backend* de PostgreSQL. Los componentes que forman parte de esta estructura interna, así como el flujo de la comunicación entre ellos al momento de procesar una consulta, se pueden observar en la Figura 6.

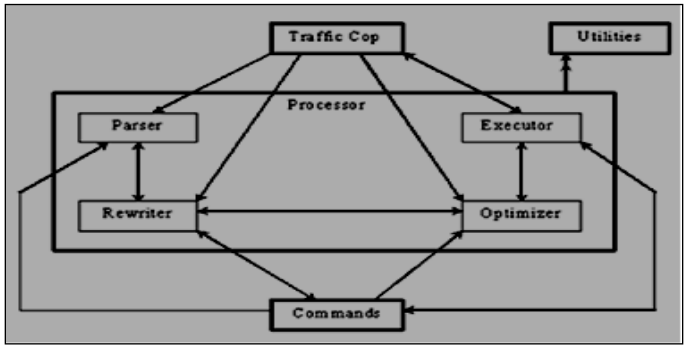

Figura 6: Arquitectura Interna del *Backend* de PostgreSQL [40]

El procesador (*processor*) está provisto de cuatro módulos: el *Parser*, el *Rewriter*, el *Optimizer* y el *Executor*. Cada una de

las tareas realizadas por estos módulos, son controladas por medio de la interacción de éstos con el *Traffic Cop*. La tarea fundamental del *Traffic Cop* (*T-Cop*) es controlar el flujo de operación de las consultas usuario en el subsistema procesador, cediendo el control operacional a cada uno de los cuatro módulos cuando corresponda.

B. *Ejecución de Operaciones en PostgreSQL*

El primer paso que se lleva a cabo para la ejecución de cualquier operación con PostgreSQL es establecer la conexión con el servidor. La misma es realizada entre el *Postmaster* y la aplicación cliente. La tarea principal de la aplicación es transmitir sentencias SQL del usuario al servidor, el cual se encargará de enviar de regreso los resultados obtenidos luego de la ejecución de dicha sentencia.

La sentencia SQL es enviada al módulo de *Parser* dentro del procesador para dar inicio a su ejecución. El *Parser* realiza el análisis gramatical y la transformación de la sentencia a una cadena de texto plano para validar su sintaxis. Luego de la validación se genera un Árbol de Traducción (*parser tree*), en caso de que la cadena sea válida, ó un mensaje de error en caso contrario. El *Parser* transforma el Árbol de Traducción en un Árbol de Consulta (*query tree*) para evaluar la existencia de las relaciones o atributos, a los cuales hace referencia dicho árbol, en el catálogo de tablas del SGBD. Si se detecta algún nombre desconocido, se devuelve un mensaje de error y se aborta el procesamiento de la sentencia. En caso contrario, el Árbol de Consulta se envía al *Rewriter*, el cual lo procesa, pues en caso de que exista alguna vista o regla que deba ser aplicada, se reescribe el árbol. El Árbol de Reescritura (*Rewritten Tree*) es enviado al *Optimizer*.

Una vez que se ha verificado que la sentencia del usuario cumple con todos los requisitos necesarios para la ejecución, el *Optimizer* refinar la ejecución, de tal manera que sea lo más económica posible para el servidor, en cuanto a consumo de recursos computacionales. Además, decide el plan de ejecución a llevar a cabo dependiendo de la estructura de la consulta. El *Executor* procesa dicho plan y se retornan las tuplas que cumplen con la condición de la consulta, o NULL cuando no existen tuplas que la cumplen.

V. Arquitectura Propuesta

La arquitectura de PostgreSQL sirve como punto de partida para construir una propuesta de arquitectura donde se incorpore al SGBD las capacidades para la gestión de consultas con minería de datos descriptiva difusa.

La propuesta arquitectónica aquí expuesta, describe el sistema donde se extiende el gestor PostgreSQLf incorporando al SQLf nuevas primitivas que añaden la semántica formal para soportar análisis de asociación y de agrupamiento difuso, de manera eficiente y escalable. Como se muestra en la Figura 7 de la Arquitectura Simplificada de PostgreSQLf, se incorporan las características novedosas directamente en los componentes ya existentes en el SGBD original.

Sin embargo, con el objetivo de garantizar las propiedades de calidad requeridas para PostgreSQLf con facilidades de minería de datos descriptiva difusa, esta propuesta de diseño arquitectónico se basa en una Arquitectura Inicial orientada a metas, aspectos y calidad. Las cualidades de calidad requeridas son rendimiento y escalabilidad, pues éstas garantizan una correcta gestión de consultas novedosas sobre grandes volúmenes de datos. En la propuesta solo se menciona el rendimiento pues la escalabilidad está incluida como parte de este requisito dentro de la norma ISO/IEC 25010.

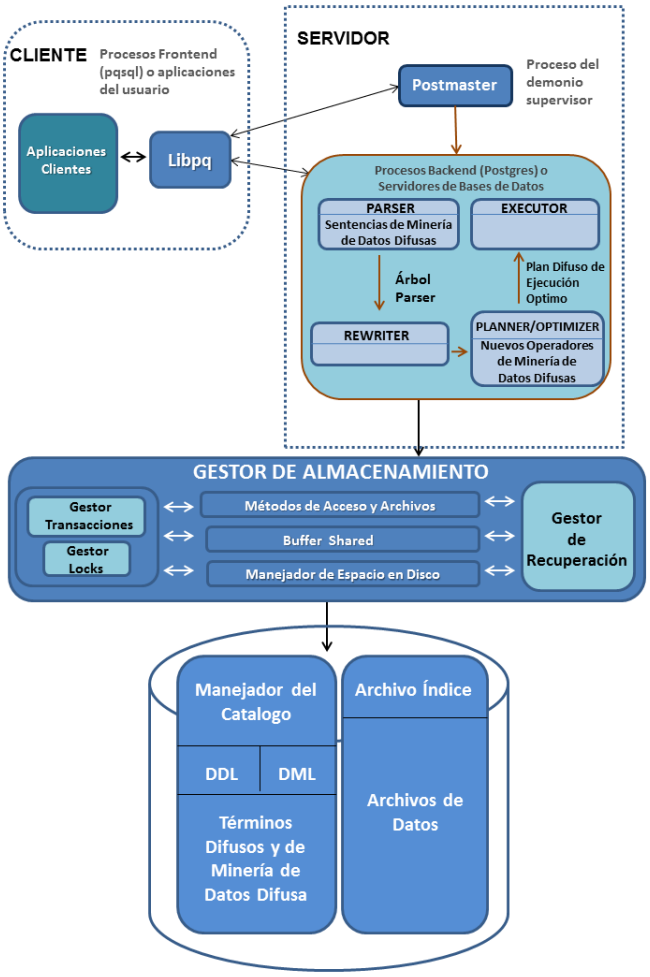

Figura 7: Arquitectura Simplificada de PostgreSQLf

Por otro lado, la gestión de grandes volúmenes de datos requiere que el diseño arquitectónico para tales capacidades se base en una estrategia de acoplamiento fuerte. Como ya se ha mencionado este tipo de acoplamiento garantiza tanto rendimiento como escalabilidad.

A continuación, se detalla la arquitectura propuesta, partiendo de una breve explicación del modelo de dominio que contiene las funcionalidades novedosas del sistema. Luego se explican las ventajas del enfoque de acoplamiento propuesto, para finalmente mostrar los resultados de la aplicación del proceso DAOMAC y de la Arquitectura Inicial obtenida.

A. *Modelo del Dominio*

Para ilustrar las funcionalidades novedosas que se desean incorporar a un SGBD, se describe el modelo del dominio a través de un ejemplo de aplicación: un sistema para la prevención de la deserción estudiantil en las universidades venezolanas.

Suponga que se tiene una tabla con los atributos de los ESTUDIANTES (ci, nombre, apellido, edad, EstratoSocial, carrera, PromedioNotas, nivel, estatus). Se quiere generar las

reglas de asociación difusas cuyo tamaño de regla sea igual a 4, con un soporte mínimo de 1 y una confianza mínima del 80%. Primero se obtienen todos los posibles subconjuntos de asociación conocidos como *itemsets*, los cuales quedan almacenados en la tabla ASSOCIATIONS. Para ello, se usa la sentencia SQL extendida:

SELECT nombre, apellido, edad, EstratoSocial, count(*) AS soporte INTO ASSOCIATIONS FROM ESTUDIANTES;

Posteriormente, tomando la tabla ASSOCIATIONS, se generan las reglas de asociación difusa con el nuevo operador propuesto DESCRIBE_FUZZY_ASSOCIATION_RULES, al cual se le indica el tamaño, el soporte y la confianza deseados:

SELECT * FROM DESCRIBE_FUZZY_ASSOCIATION_ RULES ('ASSOCIATIONS', 4, 1, 80);

Con estas reglas, si se desea contar la cantidad de estudiantes *pobres* que son *jóvenes*, *maduros* y *viejos*, cual se usa la sentencia extendida de bloque simple:

SELECT label(edad), count(*) FROM ESTUDIANTES WHERE EstratoSocial=pobre GROUP BY label(edad) USING PARTITION p(edad): {*joven*, *maduro*, *viejo*};

En esta consulta EstratoSocial=*pobre* es una condición difusa, donde pobre es un predicado difuso. Además, se tienen las etiquetas lingüísticas (*joven*, *maduro* y *viejo*) que generan una partición difusa del conjunto original de estudiantes para el atributo edad. De esta forma se generan los grupos difusos.

Aquí se observa que entre los requisitos funcionales "novedosos" se espera proveer consultas con capacidad de minería de datos descriptiva difusa, a fin de extraer conocimiento con representación en forma de reglas de asociación y de grupos difusos.

A partir de estos requisitos funcionales novedosos se infiere que los RNF esperados son: reusabilidad de los componentes de PostgreSQL, funcionamiento adecuado y preciso que induzca la corrección y pertinencia del sistema (idoneidad funcional) y finalmente, el rendimiento del SGBD en términos de velocidad de respuesta y uso de los recursos.

B. Enfoque de Acoplamiento

A manera general, el enfoque de integración usando una arquitectura fuertemente acoplada, es más sencillo debido a la totalidad de las tareas, componentes y funcionalidades a integrar, se implementan como operaciones primitivas. De esta forma, el SGBD podrá realizar las nuevas capacidades sin necesidad de conectarse o intercambiar datos con componentes adicionales, lo cual ocurre con los otros tipos de acoplamiento.

La ventaja principal de este enfoque es que se resuelven todos los problemas de escalabilidad y rendimiento presentados en los enfoques de integración para las arquitecturas débil y medianamente acopladas. Por otra parte, la limitación más relevante de la integración con una arquitectura fuertemente acoplada es la dificultad que representa mantener actualizadas oportunamente las nuevas funcionalidades que se hayan agregado al SGBD, cuando surjan nuevas versiones de éste o cuando ocurran avances teóricos en dichas funcionalidades. Esto siempre va a requerir cambios importantes en la estructura interna del código del SGBD.

C. Integración del Proceso DAOMAC

El proceso DAOMAC, descrito en la sección III, fue aplicado para extender la arquitectura general de PostgreSQLf con capacidades de Minería de Datos Descriptiva Difusa (MDDD), con el fin de construir su Arquitectura Inicial Orientada a Metas, Aspectos y Calidad.

La propuesta trata de abordar la implementación de consultas SQLf que permitan realizar un análisis descriptivo no supervisado para extraer el modelo de conocimiento con representación difusa, es decir, grupos y reglas de asociación difusas. Estas consultas se ejecutarán en el entorno de PostgreSQLf sobre los datos precisos.

Aplicando el proceso DAOMAC y partiendo del modelo del dominio descrito anteriormente, la Meta Funcional Primaria del Sistema es: Implementar (escribir, compilar y ejecutar) consultas en SQLf sobre bases de datos objeto relacionales para extraer conocimiento con representación difusa en forma de grupos y reglas de asociación, directo desde el entorno de PostgreSQLf.

Este proceso parte del modelo de negocio, en donde se describen los requisitos funcionales de minería de datos difusa. Se creó el *Pool o actor Usuario* con su respectivo *Lanes o rol de Analista de Datos*. Los pasos del proceso asociado a la meta funcional primaria se resumen en: establecer los requisitos funcionales de minería de datos difusa, selección y limpieza del conjunto de datos, selección del algoritmo de minería de datos que se va a aplicar, construcción de la vista minable, escribir las sentencias de SQLf, ejecutar la consulta y mostrar los resultados. El MN en estudio se muestra en la Figura 8, donde se observan las tareas funcionales y no funcionales necesarias para el PostgreSQLf.

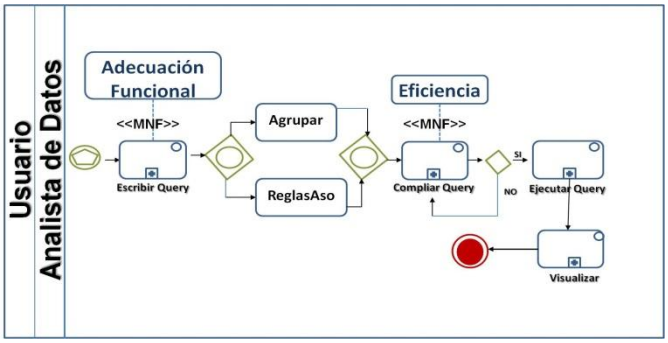

Figura 8: Modelo de Negocio de PostgreSQLf con MDDD

Este modelo de negocio está dirigido por el Modelo de Calidad del Sistema PostgreSQLf que se muestra en Figura 9. Las propiedades de calidad del modelo de calidad ISO/IEC 25010, corresponden con los RNF identificados en el modelo del dominio, donde se observó que la Reusabilidad, el Funcionamiento adecuado y el Rendimiento del sistema gestor son las cualidades no funcionales mínimas prioritarias.

A partir del MN y siguiendo las disciplinas del DAOMAC, se aplicaron una serie de transformaciones cumpliendo las reglas de correspondencia semántica entre BPMN y GRL [23], descritas en la Figura 10.

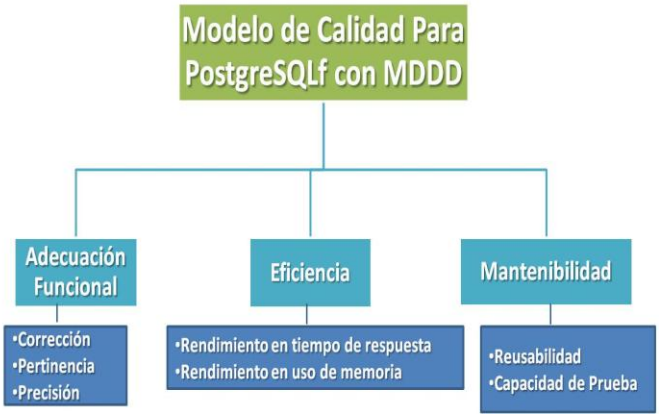

Figura 9: Adaptación del Modelo de Calidad ISO/IEC 25010 para PostgreSQLf con MDDD

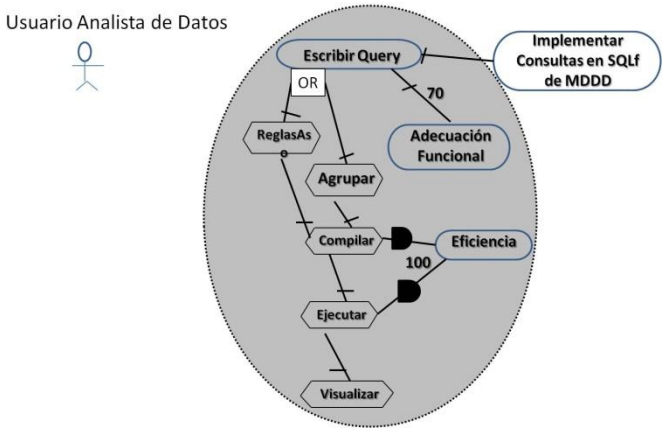

Figura 11: Modelo de Metas de PostgreSQLf en GRL

Una vez aplicadas las transformaciones se obtuvo el modelo de metas, mostrado en la Figura 11, para el *Pool* Usuario en su rol de Analista de Datos, correspondiente a la meta principal del MN, denominado Modelo de Metas Implementar Consultas para MDDD en SQLf. Se detectaron las Metas No Funcionales que estén asociadas a aspectos no funcionales del negocio en BPMN y se asocian con las Metas No Funcionales o *softgoals* de GRL, por lo qu e se asoció la tarea funcional EscribirQuery a la «MNF» *Adecuación Funcional* así como la «MNF» *Eficiencia* a la tarea funcional *Compilar*.

Luego el Modelo SIG se construyó para la Meta No Funcional Transversal Principal *Eficiencia*, el cual se observa en la Figura 12. En primer lugar se identificó las metas funcionals y no funcionales transversales en el modelo de metas anterior para el actor Usuario en su rol de Analista de Datos; se puede notar que la adecuación Funcional no es una Meta No Funcional Transversal ya que solo entrecruza a la Meta Funcional EscribirQuery. Adicionalmente se asignaron prioridades a las metas no funcionales, donde se evidenció que la Eficiencia tiene alta importancia, en vista que para el sistema que el tiempo de respuesta y uso de memoria es fundamental para implementar las consultas de mineria descriptiva con grandes

Regla	Lenguaje BPMN			Lenguaje GRL			Regla para la Correspondencia
	Termino	Definición	Notación Gráfica	Termino	Definición	Notación Gráfica	
1	*Process* o Proceso	Actividad realizada dentro o a través de empresas u organizaciones [48].	Texto	*Hardgoal* o Meta Funcional	Condición o estado de un asunto que a los *stakeholders* les gustaría lograr [30].		Un *Proceso* de BPMN representa una *Meta Funcional* en GRL.
2	*Subprocess* o Subproceso	Agregación de actividades que son incluidas dentro de un Proceso [48].					Un *Subproceso* de BPMN representa una *Meta Funcional* que ha sido descompuesta en GRL en otras metas de nivel inferior
3	*Pool*	Participantes (entidades o roles de negocio) [48].		*Actor*	Entidades activas que llevan a cabo ciertas acciones para lograr metas [30].	Actor	Un *Pool* de BPMN representa un *Actor* en GRL.
4	*Lane* o Carril	Organizan y categorizan actividades de un *Pool* [48].		*Role* o Papel	Caracterización abstracta del comportamiento de un actor [1].	Actor «Rol»	Un *Carril* de BPMN es descrito como el *Papel* de un Actor concreto en GRL.
5	*Task* o Tarea	Es una actividad atómica [48].	Tarea 1	*Task* o Tarea	Es una manera particular de hacer algo [30].		Una *Tarea* de BPMN representa una *Tarea* particular en GRL.
6	Labeled Association o Asociación Etiquetada	Muestra la asociación entre una tarea o subproceso funcional y una Meta No funcional, «MNF»	«MNF»	*Softgoal* o Meta No Funcional	Es una condición o estado de un asunto del mundo que a los *stakeholders* les gustaría lograr (ITU-T, 2012).		Una Asociación Etiquetada *de* BPMN relaciona una tarea funcional o subproceso de BPMN con *un Softgoal* en GRL.
7	*Data Object* u Objeto de Datos	Artefacto que suministra información a las actividades a ser realizadas [48].		*Resource* o Recurso	Entidad física o informacional que expresa necesidades [30].		Un *Objeto de Datos* en BPMN representa un *Recurso* de GRL.
8	*Gateway* o Compuerta	Controlan la interacción, convergencia o divergencia dentro de un proceso [48].		*Task decomposition link* o Vínculo de descomposición de tarea	Tipo de vínculo que descompone una tarea en un *hardgoal, subtask, resource* o *softgoal* [30].		Una *Compuerta* en BPMN representa un *Vínculo de descomposición de tareas* (And/Xor/Ior) de GRL.
9	*Sequence Flow* o Flujo de Secuencia	Muestra la secuencia de actividades que son realizadas en un proceso [48].		*Means-Ends link* o Vínculos Medio-Fin	Relación binaria entre un fin y el medio para lograrlos. El "medio" representa una tarea y el "fin" una meta [30].		Un *Flujo de Secuencia* en BPMN representa un *Vínculo Medio-Fin* en GRL.
10	*Message Flow* o Flujo de Mensaje	Muestra el flujo de mensaje entre dos entidades [48].		*Dependency link* o Vinculo de dependencia	Relación intencional entre dos actores [30].		Un *Flujo de Mensaje* en BPMN representa un *Vínculo de Dependencia* entre dos actores en GRL.

Figura 10: Reglas de Correspondencia Semántica entre BPMN y GRL [23][39]

volúmenes de datos.

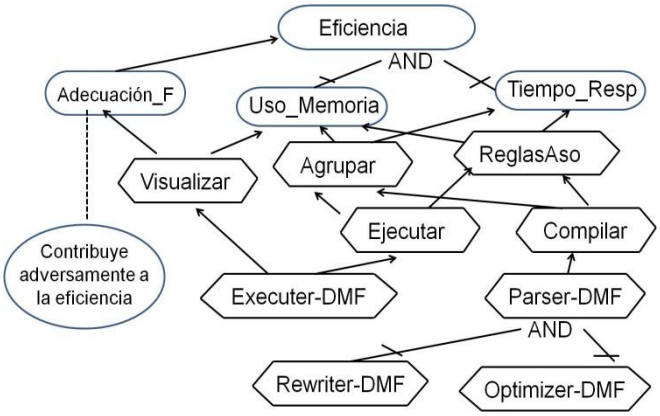

Figura 12: Modelo SIG para la Meta No Funcional Transversal Eficiencia

Finalmente, a partir de estos modelos se construye la Arquitectura Inicial orientada a metas, aspecto y calidad del Sistema PostgreSQLf con capacidades de Minería de Datos Difusa. En ella, las metas funcionales y las metas no funcionales transversales del Modelo de Metas, corresponden a Componentes Funcionales y Componentes de Aspectos (propiedades de calidad), respectivamente.

En la Figura 13, se muestra la AI, orientada a metas, aspectos y calidad, propuesta para PostgreSQLf. Ésta se obtuvo luego de aplicar el proceso de diseño arquitectónico DAOMAC, habiendóse aplicado técnicas orientada a metas, aspecto y calidad. La particularidad, de esta propuesta arquitectónica, son los componentes de aspecto relativos a las propiedades de calidad («Aspect») del sistema, denominados Eficiencia y Adecuación_Funcional, que corresponden a las metas no funcionales transversales del modelo de metas, que surgieron de los requisitos no funcionales. Los demás componentes, los cuales no son etiquetados con «Aspect», se refieren a las metas funcionales del sistema. Éstos pueden ser mecanismos, módulos, bases de datos, programas ejecutables, entre otros, que representan una solución arquitectónica para la meta funcional planteada.

Los círculos azules indican las interfaces, lazos de unión entre varios componentes ("*provide/require*"), reflejan las interacciones entre los diferentes componentes del diagrama. La existencia de una conexión muestra que un componente hace uso de la interfaz de otro, posiblemente a través de una solicitud de servicio, envío de un mensaje o comunicación de información.

La Adecuación Funcional sólo aparece conectada con los componentes Visualizar, Escribir_Query_MDDD y Executor_MDDD, pues este requisito no funcional o meta, sólo es asociado a las metas funcionales correspondientes a estos componentes. Este requisito se refiere a la completitud, pertinencia, precisión y corrección funcional durante el uso del producto de software. En este caso particular, dicha meta es importante para estos componentes pues corresponde a la especificación de las consultas y a la visualización de resultados asociados a la extensión MDDD. Así como también, a la ejecución de tales consultas. La complejidad intrínseca a este tipo de consultas demanda interfaces adecuadas que faciliten estos procesos. A través de estas interfaces, el usuario debe ser capaz de entender y gestionar las capacidades

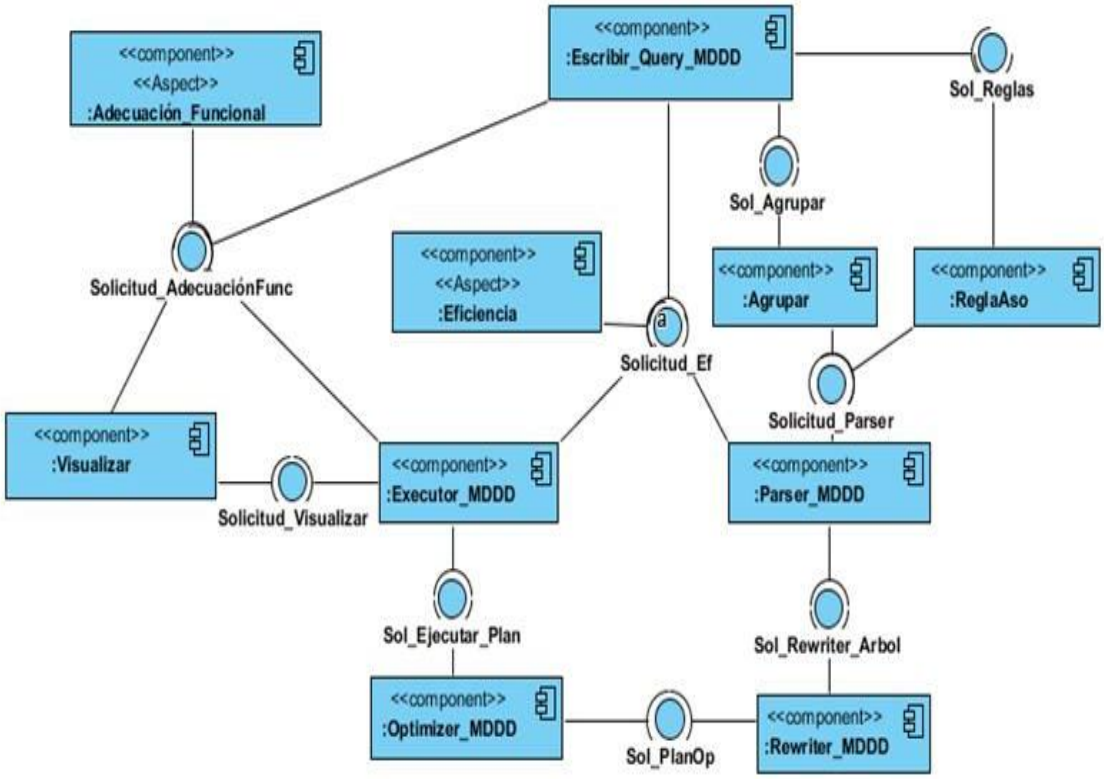

Figura 13: Arquitectura Inicial de PostgreSQLf en UML

vinculadas, a la minería de datos descriptiva difusa, que se hacen sobre los datos.

Durante la escritura de la consulta, por ejemplo, el conjunto de datos debe configurarse para poder aplicar los algoritmos de minería de datos difusa. El componente Escribir_Query_MDDD es entonces el ambiente para construir la sentencia a ser enviada a los demás componentes funcionales. Aquí, la eficiencia tiene que ver con el tiempo de respuesta y el uso de la memoria en la interfaz gráfica. Por otro lado, Visualizar es el componente que se refiere a la interfaz para observar el resultado final de una consulta MDDD. La eficiencia es un requisito de mayor importancia durante el análisis y ejecución de las consultas.

Los componentes Executor, Parser, Rewriter, Optimizer tienen una correspondencia directa con la estructura interna de PostgreSQLf (ver Figura 7), a los cuales se le agrega el sufijo "_MDDD", para indicar que han sido extendidos con capacidades de minería de datos descriptiva difusa.

ReglasAso es el componente que permite configurar una consulta que se va analizar usando reglas de asociación difusas. Mientras que Agrupar, permite configurar las consultas con agrupamiento difuso. En ambos se deben proveer mecanismos para especificar el tipo de algoritmo a usar durante la ejecución.

VI. Conclusiones y Trabajos Futuros

Se ha propuesto una arquitectura inicial o AI para el Sistema PostgreSQLf con capacidades de Minería de Datos Descriptiva Difusa, orientada a metas, aspectos y calidad, utilizando el modelo de diseño arquitectónico DAOMAC.

La AI construida considera el RNF de calidad Eficiencia, en términos de uso de recursos y tiempo de respuesta. Este requisito es un atributo de calidad primario (prioritario), basado en ISO/IEC 25010, relacionado con la meta funcional primaria de implementar consultas en SQLf para extraer grupos y reglas de asociación difusas, usando datos precisos almacenados en un Sistema Gestor de Bases de Datos.

Se propone extender el lenguaje SQLf con operadores para extraer reglas y grupos difusos, implementados en PostgreSQLf usando una arquitectura de acoplamiento fuerte que está basada en la arquitectura de referencia de PostgreSQL.

Se aplica el proceso de diseño arquitectónico DAOMAC, para la MNF de Eficiencia, con un resultado alentador. Se construye una Arquitectura Inicial y este resultado intermedio muestran la factibilidad de su aplicación.

Con esta arquitectura se espera beneficiar al proceso de análisis de asociación y de agrupamiento difuso con el rendimiento y escalabilidad deseados, considerando estos requisitos no funcionales desde altos niveles de abstracción como el Modelo de Negocio.

Como trabajos futuros, se han plateados los siguientes objetivos. Primero, refinar la Arquitectura Inicial construida hasta "valorar" la trazabilidad de los atributos de calidad propuestos desde el Modelo de Negocio. Segundo, aplicar la arquitectura para la totalidad de los requisitos funcionales y no funcionales del Sistema PostgreSQLf con facilidades de Minería de Datos Descriptiva Difusa. Tercero, comparar los resultados de aplicar la arquitectura a diversos casos de estudio que requieran otras funcionalidades novedosas.

Agradecimientos

Son muchas las personas que realizaron aportes de vital importancia para el logro de los objetivos planteados en el presente trabajo. En especial mención, extendemos nuestros agradecimientos al Dr. Leonid Tineo, Profesor Titular de la Universidad Simón Bolívar, Venezuela, por sus consejos y consideraciones técnicas ofrecidas oportunamente. Finalmente, agradecemos infinitamente al dador de toda ciencia y conocimiento, ¡El Señor creador de la vida! Quién nos ha dado los recursos y las fuerzas.

Referencias

[1] P. Bosc and O. Pivert, *SQLF: A Relational Database Language for Fuzzy Querying*. IEEE Transactions on Fuzzy System, vol. 3, no. 1, pp. 1-17, 1995.

[2] L. Tineo, *Extending RDBMS for Allowing Fuzzy Quantified Queries*. Lecture Notes in Computer Science, vol. 1873, pp. 407-416, 2000.

[3] R. Timarán, *Nuevas Primitivas SQL para el Descubrimiento de Conocimiento en Arquitecturas Fuertemente Acopladas con un Sistema de Gestion de Bases de Datos*. Tesis Doctoral. Universidad del Valle, Santiago de Cali, Colombia, 2005.

[4] P. Bosc and O. Pivert, *SQLf Query Functionality on Top of a Regular Relational Database Management System*. In: Knowledge Management in Fuzzy Databases, O. Pons, M. A. Vila, J. Kacprzyk (eds.), pp. 171-190, 2000.

[5] A. Aguilera, L. Borjas, R. Rodríguez, and L. Tineo, *Experiences on Fuzzy DBMS: Implementation and Use*, in proceedings of the XXXIX Latin American Computing Conference (CLEI 2013), vol. 1, pp. 478-485, Naiguatá, Venezuela, Octubre 2013.

[6] M. Goncalves and L. Tineo, *SQLf3: An Extension of SQLf with SQL3 Features*, in proceedings of the 10th IEEE International Conference on Fuzzy Systems (Fuzzy IEEE), Melbourne, Australia, December 2001.

[7] J. Ramírez and L. Tineo, *Un Mecanismo de Respuestas a Consultas en Presencia de Nulos*, Revista Venezolana de Computación (ReVeCom), ISSN: 2244-7040, vol. 2, no. 1, pp. 48-59, Diciembre 2015.

[8] A. Aguilera, L. Borjas, and R. Rodríguez, *Un Modelo de Proceso Heurístico para Implementación de SGDB Difusos*, Memorias de la Tercera Conferencia Nacional de Computación, Informática y Sistemas, ISBN: 978-980-7683-01-2, pp. 242-247, Valencia, Venezuela, Octubre 2015.

[9] L. A. Zadeh, *Fuzzy Sets*, Information and Control, vol. 8, 1965.

[10] L. Yan, Z. Ma, and F. Zhang, *Fuzzy Sets and Fuzzy Database Models*, Fuzzy XML Data Management, pp. 31-81, Springer Berlin Heidelberg, 2014.

[11] C. Zapata, C. Jiménez, and J. Velásquez, *El Tratamiento de los Terminos Borrosos en Manejadores de Base de Datos Relacionales*, DYNA, vol. 70, no. 138, pp. 1-11, 2003.

[12] A. Urrutia, J. Galindo, and A. Sepúlveda, *Implementación de una Base de Datos Difusa con FIRST-2 y PostgreSQL*, XV Congreso Español sobre Tecnologías y Lógica Fuzzy, ESTYLF, pp. 199-204, Huelva, España, Febrero 2010.

[13] P. Bosc and O. Pivert, *On the Eficiency of the Alpha-cut Distribution Method to Evaluate Simple Fuzzy Relational Queries*, Advances in Fuzzy Systems-Applications and Theory, Fuzzy Logic and Soft Computing, Bouchon-Meunier, pp. 251-260, 1995.

[14] P. Bosc and O. Pivert, *On a Fuzzy Group-by Clause in SQLf*, in proceeding of the IEEE International Conference on Fuzzy Systems, pp 1-6, Barcelona, España, July 2010.

[15] R. Sabatino, *PostgreSQLf: Extensiones de Cuantificadores Difusos y Agrupamiento Difuso sobre el SGBD PostgreSQL*, Trabajo de Grado, Universidad de Carabobo, Valencia, 2010.

[16] A. Aguilera, J. Cadenas, and L. Tineo, *Fuzzy Querying Capability at Core of a RDBMS*, in Advanced Database Query Systems: Techniques, Applications and Technologies, L. Yan and Z. Ma (eds.), IGI Global. New York, USA, 2011, pp. 160-184.

[17] D. Garlan and M. Shaw, *Software Architecture: Perspectives on an Emerging Discipline*, Prentice Hall, 1996.

[18] C. Ghezzi, M. Jazayeri, and D. Mandrioli, *Fundamentals of Software Engineering*, Prentice Hall, 2002.

[19] IEEE, *IEEE Standard 1471-2000 – IEEE Recommended Practice for Architectural Description for Software-Intensive Systems, 2000*.

[20] F. Losavio, L. Chirinos, N. Lévy, and A. Ramdane-Cherif, *Quality Characteristics for Software Architecture*, Journal of Object Technology, vol. 2, no. 2, pp. 133-150, 2003.

[21] F. Losavio and Ch. Guillén. *Marco Conceptual para un Diseño Arquitectónico basado en Aspectos de Calidad*, Revista Universitaria Sapiens, vol. 7, no. 2, pp. 119-138, 2006.

[22] J. C. Guzmán, F. Losavio, and A. Matteo, *Comparación de Métodos para el Diseño Arquitectónico del Software que Consideran Metas, Aspectos y Estándares de Calidad*, Enl@ce: Revista Venezolana de Información, Tecnología y Conocimiento, vol. 10, no. 2, pp. 11-27, 2013.

[23] J. C. Guzmán, F. Losavio, and A. Matteo, *Del Modelo de Negocio a la Arquitectura del Sistema Considerando Metas, Aspectos y Estándares de Calidad*, Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS), vol. 3, no. 2, pp. 19-37, 2013.

[24] M. Mlitat and Z. Dai, *ISO/IEC 25010: Software Engineering-Software product Quality Requirements and Evaluation (SQuaRE)*, Quality models, 2011.

[25] M. Hammer and J. Champy, *Reengineering the Corporation: A Manifesto for Business Revolution*, Harper Collins, 1993.

[26] M. Hepp and D. Roman, *An Ontology Framework for Semantic Business Process Management*, in proceeding of the 8th International Conference Wirtschafts Informatik, pp. 1-18, Karlsruhe, Germany, 2007.

[27] A. Bluter, *Business Process Management Essentials*, Glintech, 2009.

[28] S. White, *Business Process Modeling Notation* (*BPMN*) 1.0. Business Process Management Initiative. 2004.

[29] P. Krutchen, *The Rational Unified Process: An Introduction*. Addison Wesley, 2000.

[30] T. Crusson, *Business Process Modeling Languag*, GLiNTECH, 2006.

[31] OMG. *Business Process Model and Notation* (*BPMN*) 2.0, http://www.omg.org/spec/BPMN/2.0. 2011.

[32] ITU-T, *ITU-T Z.151: User Requirements Notation* (*URN*)-*Language Definition, Recommendation*, 2012.

[33] OMG, *Software Process Engineering Metamodel* (*SPEM*) 2.0, http://www.omg.org/spec/SPEM/2.0/PDF. 2008.

[34] OMG, *Unified Modeling Language*™ (*UML*®) 2.0, http://www.omg. org/spec/UML, 2005.

[35] L. Chung, B. Nixon, and E. Yu, *Using Non-Functional Requirements to Systematically Select Among Alternatives in Architectural Design*, in proceeding of 1st Int. Workshop on Architectures for Software System, pp. 31-32, Washington, USA, 1995.

[36] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulus, *Non-Functional Requirements in Software Engineering*, International Series in Software Engineering, Springer, 2000.

[37] A. Lamsweerde, *Goal-Oriented Requirements Engineering: A Guided Tour*, in proceeding of the 5th Intl. Symp. Req. Eng. (RE'01), pp. 249-263, Toronto, Canada, August 2001.

[38] L. Chung, K. Cooper, and A. Yi, *Developing Adaptable Software Architectures Using Design Patterns: a NFR Approach*, Journal Computer Standards & Interfaces - 36 Special issue: Adaptable software architectures, vol. 25, no. 3, pp. 253-260, 2002.

[39] F. Losavio, J. C. Guzmán, and A. Matteo, *Correspondencia Semántica entre los Lenguajes BPMN y GRL*, Enl@ce Revista Venezolana de Información, Tecnología y Conocimiento, vol. 8, no. 1, pp. 11-29, 2011.

[40] H. Chen, H. Lim, and J. Xiao, (s.f.). *Concept architecture of PostgreSQL*.

[41] F. Yuan, J. She, Z. Fan, and J. J. Wu, *Concrete Architecture of PostgreSQL Backend*, http://www.cs.uwaterloo.ca/ f2yuan.