Benchmarking de Consultas Difusas utilizando PostgreSQL

J.T. Cadenas*[1,2] A. Aguilera[2] y L. Tineo[1,2]

[1]Universidad Simón Bolívar, Departamento de Computación y T.I., Caracas, Venezuela
[2]CAMYTD, Centro de Análisis, Modelado y Tratamiento de Datos, FACYT, Universidad de Carabobo, Valencia, Venezuela

RESUMEN

La comunidad de base de datos se resiste a adoptar nuevas herramientas para extender las funcionalidades de un Sistema Gestor de Bases de Datos Relacionales (RDBMS) porque normalmente se añade tiempo de procesamiento que degrada el rendimiento y afecta la escalabilidad del sistema. En este artículo se propone una comparación del rendimiento (Benchmarking) de consultas difusas implementadas utilizando el RDBMS PostgreSQL comparado con consultas clásicas. Se diseñaron experimentos computacionales basados en modelos estadísticos de análisis de varianza para múltiples factores para verificar su validez. Se demuestra que el tiempo de ejecución de las consultas con una estrategia de implementación de acoplamiento débil es estadísticamente mayor que al utilizar una de acoplamiento fuerte; además, no hay diferencia estadística significativa entre las consultas difusas con una estrategia de implementación acoplamiento fuerte y sus respectivas consultas clásicas. Finalmente se comprueba que la estrategia débilmente acoplada no escala.

Palabras Clave: Consultas Difusas, PostgreSQL, SQLf, RDBMS

ABSTRACT

The database community has resisted adopting new tools that offer flexibility to extend the facilities of a Relational Database Management System (RDBMS), this is due the undesired processing time affecting the system performance and scalability. In this article we achieve a benchmarking of fuzzy queries to measure the performance and scalability of diverse implementation strategies versus classical queries using the RDBMS PostgreSQL. We designed a series of computational experiments based on a statically analysis of variance for multiple factors to verify their validity. We proof that the elapsed time of queries with a strategy of loose coupling is statically higher than than that obtained with a tight coupling; moreover, there are no statically difference between tight coupling and classical queries. Finally, we proof that loose coupling strategy doesn't have scalability.

Keywords: Fuzzy queries, PostgreSQL, SQLf, RDBMS

1. Introducción

En la actualidad hay un gran interés en la comunidad científica de bases de datos en la gestión de la incertidumbre; es así como se han desarrollado diversas propuestas tales como: MystiQ en la Universidad de Washington [BDM*05], el cual es un sistema que usa un modelo de datos probabilístico para encontrar respuestas en un gran número de fuentes de datos exhibiendo varios tipos de imprecisiones; Trio, creado en la Universidad de Stanford [Wid09], como un Sistema Gestor de Base de Datos donde el dato, la incertidumbre y el linaje (procedencia del dato) son gestionados en forma integrada; MayBMS [Koc09] de la Universidad de Cornell es un sistema para la administración escalable de información incierta (utilizando un modelo probabilístico) implementado sobre PostgreSQL.

En ocasiones no sólo la incertidumbre de la información sino la imprecisión en la búsqueda de información lo que hace necesario la incorporación de mecanismos adicionales tales como la lógica difusa [Zad94].

Es así como desde hace varios años diversos investigadores han modelado y gestionado la incertidumbre en los RDBMS utilizando la teoría de conjuntos difusos [Zad65] y la lógica difusa dando lugar a un área de investigación denominada Bases de Datos Difusas, que incluye: modelado y representación de datos imprecisos, además de la utilización de condiciones flexibles en las consultas a Bases de Datos Clásicas, a lo que se denomina consultas difusas. Para el lector interesado en las diversas propuestas existentes Galindo [Gal08] presenta un compendio de trabajos recientes; la mayoría de estas propuestas están desarrolladas para ser realizadas en Sistemas Gestores de Bases de Datos Relacionales (RDBMS).

El adecuado rendimiento de los Sistemas Gestores de Bases de datos es una condición necesaria para ser aceptados [CB05]; sin embargo, este problema ha sido relegado a un segundo plano en las investigaciones sobre sistemas gestores de bases de datos difusas [LT06]. En este artículo se demuestra cómo mejorar el rendimiento y escalabilidad de consultas difusas, implementando SQLf [BP00], para

proporcionar flexibilidad y manejo de preferencias mediante consultas difusas efectuadas en el RDBMS PostgreSQL.

El artículo se organiza de la siguiente forma: en primer lugar se presenta un Marco Teórico, luego el diseño experimental, seguido del resultado de los experimentos, para finalmente presentar las conclusiones y trabajos futuros.

2. Marco Teórico

A continuación se describe un estudio del estado del arte de Bases de Datos Difusas, el lenguaje de consultas difusas SQLf y diversas estrategias para su implementación en un RDBMS.

2.1 Estado del Arte de Bases de Datos Difusas

Una aplicación de consultas difusas como una herramienta para la toma de decisiones se presenta en el artículo [AR11]. Para ello se integran los datos del Laboratorio de Marcha del Hospital Ortopédico Infantil en una base de datos que contiene información multimedia proveniente de los exámenes complementarios (señales y radiografías), del examen físico articular y del video de marcha de los pacientes. Las consultas difusas permiten incorporar los beneficios de la lógica difusa para expresar requerimientos que involucran preferencias de usuarios y consultas que sean más cercanas al razonamiento humano.

Un artículo de revisión de modelos de datos conceptuales difusos propuestos en la literatura se encuentra en [MY10]; se discuten principalmente los modelos de datos difusos ER/FER, IFO y UML; además se examinan las aplicaciones de los modelos de datos conceptuales difusos. Los autores afirman que la teoría de conjuntos difusos ha sido ampliamente aplicada para extender diversos modelos de bases de datos, lo que ha dado lugar a numerosas contribuciones, principalmente con el popular modelo relacional o con alguna forma vinculada al mismo. Además justifican el uso de modelos de datos semánticos difusos debido a que múltiples investigadores se han concentrado recientemente en él para satisfacer la necesidad de modelar objetos complejos con imprecisión e incertidumbre.

En [UGS10] se presenta una extensión de una base de datos relacional difusa utilizando como estructura lógica FIRST-2 (una interfaz difusa para sistemas relacionales), desarrollada sobre PostgreSQL. Además, presentan una aplicación a un sistema de citas médicas.

Un *framework* para consultar bases de datos es presentado en [Yag10]. El autor modela una consulta como una colección de condiciones requeridas y un imperativo para combinar una satisfacción del objeto a las condiciones individuales para obtener una satisfacción total. El autor investiga herramientas que pueden enriquecer este proceso habilitando la inclusión de consideraciones más centradas en el ser humano; discute cómo incluir condiciones flexibles con el uso de conjuntos difusos. Se describen técnicas algo más sofisticadas para agregar las satisfacciones de condiciones individuales basadas en la inclusión de importancias y el uso del operador OWA. Se introduce un nuevo método para agregar las satisfacciones individuales que pueden modelar una relación lexicográfica entre los requerimientos individuales; además de considerar aspectos más centrados en el humano para la consulta, el autor busca en las bases de datos en la cual la información puede tener alguna incertidumbre posibilística o probabilística.

Aunque las bases de datos relacionales difusas han sido ampliamente estudiadas a nivel teórico, son pocas las repercusiones en la práctica según lo afirmado en [GB09]. Los autores proponen una nueva arquitectura para sistemas gestores de bases de datos difusas (FRDBMS) extendiendo el modelo GEFRED propuesto por Medina, Pons y Vila [MPV94] mediante una estrategia de implementación de acoplamiento débil sobre Oracle; permitiendo que el usuario directamente efectúe la descripción, manipulación y consultas utilizando el lenguaje FSQL. Ellos alegan que la limitación de GEFRED y otras propuestas como la de Bosc y Pivert [BP95], [BP00] es que el FRDBMS es implementado manualmente por el usuario.

La información a ser almacenada en las bases de datos es frecuentemente difusa según lo afirmado en [LLL09]; dos problemas importantes de investigación en este campo son la representación de información difusa en una base de datos y la provisión de flexibilidad en consulta a bases de datos, mediante la inclusión de términos lingüísticos en consultas orientadas al ser humano y regresando resultados con grados de coincidencia. La programación lógica lingüística difusa (FLLP), introduce una representación fácil y un razonamiento expresado lingüísticamente del conocimiento humano. El artículo presenta un modelo de datos basados en FLLP llamado Data log lingüístico difuso para bases de datos lingüísticas difusas con consultas flexibles.

En el artículo [RCC09] se presenta una arquitectura de base de datos difusas denominada Alianza, la cual introduce una manera de representar conceptos difusos (meta-información base utilizando formato XML) simplificando las actividades de gestión de los datos en estos ambientes. Alianza es producto de la unión de técnicas de lógica difusa, sistemas gestores de base de datos relacionales y una base de meta-conocimiento difuso (FMB) definida en XML; con el fin de manipular y representar información vaga.

Se proponen sistemas de base de datos inciertos en [Che09], donde se considera la incertidumbre en los datos, además del uso de datos genéricos y modelos para capturar incertidumbre; también se proporcionan operadores de consulta que devuelvan respuestas con confianzas estadísticas. El autor considera que la gestión de incertidumbre en grandes bases de datos ha atraído un interés tremendo de investigación. La incertidumbre de los datos es inherente en muchas aplicaciones emergentes e importantes, tales como servicios basados en localización, redes sensoriales inalámbricas, aplicaciones de flujo de datos, bases de datos biométricas y biológicas; en estos sistemas es importante gestionar los datos y la incertidumbre, a fin de tomar decisiones correctas y proporcionar servicios de alta calidad a los usuarios.

La noción de conjuntos vagos en relaciones es propuesta en [ZM09], donde se describe el SQL vago como una extensión de SQL para el modelo relacional vago. Se demuestra que SQL vago (VSQL) combina las capacidades

de un SQL estándar con el poder de manipular relaciones vagas. Los autores afirman que aunque SQL vago es una extensión mínima para ilustrar sus usos, VSQL permite a los usuarios formular un amplio rango de consultas entre datos vagos y consultas. Todo ello como una forma comprehensiva para manipular datos imprecisos e inciertos.

En su artículo [DAC*08] proponen la construcción de un sistema de información y ayuda a la toma de decisión (IDSS) que permita a distintos tipos de usuarios (agricultores, agrónomos, administraciones públicas) obtener y manipular información sobre el cultivo de olivas y el soporte ambiental del mismo para ayudar en la toma de decisiones. Los principales objetivos desarrollados son el tratamiento de datos inciertos e imprecisos de la información ambiental y sobre cultivos, además de la fusión de datos sobre cultivo y otros de carácter científico-experimental. Se describe la posibilidad de almacenar la información de carácter agronómico y ecológico en bases de datos relacionales. La información es procesada a través de herramientas de extracción de conocimiento y permite sobre la base del conocimiento experto el desarrollo de reglas para la clasificación de aptitud del terreno y para la obtención de mapas temáticos con la ayuda de Sistemas de Información Geográfica. Las consultas flexibles permiten a los distintos usuarios la consulta interactiva de toda la información almacenada en las bases de datos, así como una implementación constante de las mismas.

2.2 SQLf

El SQL es un lenguaje versátil para recuperar información sobre bases de datos relacionales existentes, pero presenta un problema de rigidez cuando el usuario no puede definir en forma precisa su consulta, sino que tiene una idea vaga de la misma o la consulta está relacionada con términos que son imprecisos por su naturaleza (una persona *joven*, un objeto *costoso*, un hotel *cercano*), por lo que ningún RDBMS clásico brinda mecanismos que puedan ayudar al usuario en la solución de tal necesidad.

Los autores Bosc y Pivert proponen SQLf [BP95] como una extensión del lenguaje SQL para incorporar el uso de la lógica difusa en las consultas a bases de datos relacionales, permitiendo efectuar consultas incorporando condiciones difusas, con el fin de permitir representar requerimientos de los usuarios de forma más cercana a la realidad.

Similar a SQL, el constructor fundamental de SQLf es el bloque básico el cual tiene la siguiente estructura:

SELECT [DISTINCT] <*lista de atributos*>
FROM <*lista de relaciones*>
WHERE <*condición difusa*>
[WITH CALIBRATION <*c*>]

La cláusula WITH CALIBRATION es opcional (al igual que DISTINCT), donde <*c*> es el *factor de calibración* que permite al usuario determinar un filtro adicional de acuerdo al grado de membresía a la consulta o un número determinado de tuplas, ya sea un número real mayor a 0 y menor o igual a 1 ($0 < c \leq 1$) o un número entero positivo mayor a 1, respectivamente

En cuanto a la condición difusa, en su forma más sencilla, es la determinación del grado de membresía en que una expresión satisface un predicado difuso definido por el usuario, utilizando variables lingüísticas (tales como edad o tamaño) y valores o etiquetas lingüistas (por ejemplo: joven, viejo, alto, bajo), definidos dentro de la teoría de lógica difusa como números difusos (conjuntos difusos que cumplen con las propiedades de normalidad y convexidad) con función de membresía.

SQLf soporta el uso de diversas estructuras para construir condiciones difusas más complejas, tales como: modificadores difusos, conectores difusos, comparadores difusos, cuantificación difusa y particionamiento con calificación de cada grupo en base a una condición difusa.

SQLf ha sido actualizado al estandar SQL 1999 [GT01a] permitiendo vaguedad en operadores de algebra, restricciones de integridad, vistas, tipos de datos hora y fecha; operaciones de manipulación de datos, reglas deductivas y aserciones; objetos, funciones, procedimientos y disparadores. También se extendió SQLf para permitir características basadas en el estándar SQL 2003 [GT01b] tales como: tipos de datos multiconjunto, extensiones a las instrucciones: *CREATE TABLE*, *MERGE* y operaciones *ROLAP* de *datawarehouse*.

2.3 Estrategias de Implementación de SQLf en un RDBMS

De acuerdo a cómo se efectúa la integración de SQLf en un RDBMS se pueden identificar tres tipos de estrategias de implementación: acoplamiento débil, medio o fuerte según lo propuesto por Timarán [Tim01].

2.3.1 Acoplamiento Débil

Esta estrategia consiste en implementar una capa lógica por encima del RDBMS. Esta capa lógica se encarga de aceptar consultas con la sintaxis de SQLf, transformarlas en consultas clásicas aplicando el principio de derivación [BP00] y luego reprocesar el resultado para devolver las tuplas resultantes con grados de membresía. La principal ventaja de esta estrategia de implementación es su portabilidad, las desventajas incluyen la degradación del rendimiento y problemas de escalabilidad. Para efectos del presente artículo se utilizó una implementación débilmente acoplada denominada SQLfi [GT08]. Esta implementación fue inicialmente elaborada para trabajar sobre el RDBMS Oracle, pero luego fue llevada a diversos RDBMS tales como: PostgreSQL, SQLServer, Firebird, MySQL y DB2. SQLfi proporciona todas las características de SQLf que incluyen las extensiones del estándar de SQL 1999.

2.3.2 Acoplamiento Medio

En esta estrategia se utiliza lenguaje nativo del DBMS, tal como el PL-SQL de Oracle. Se implementó un prototipo utilizando Prolog (SQLf/PL). El lector interesado puede revisar más detalles sobre esta implementación en [UTG08].

2.3.3 *Acoplamiento Fuerte*

Esta estrategia consiste en efectuar el procesamiento de la consulta difusa en el núcleo o *kernel* del RDBMS. Se aceptan consultas difusas con la sintaxis de SQLf (extensión del analizador de consultas o *parser*), el procesamiento se hace internamente extendiendo los operadores físicos y mecanismos de acceso del RDBMS y finalmente se devuelven filas con el grado de membresía. Esta implementación brinda mejoras en el rendimiento y escalabilidad, su principal desventaja es la carencia total de portabilidad. Se desarrolló una extensión del PostgreSQL para el procesamiento de consultas difusas utilizando esta estrategia la cual se denomina PostgreSQLf, la cual es descrita en detalle en [ACT11].

3. Diseño Experimental

Para comprobar la existencia de diferencia significativa estadísticamente [BS07] entre los tiempos de respuesta de consultas difusas a base de datos y sus respectivas consultas clásicas a continuación se describe el diseño experimental y la metodología seguida.

Para las consultas difusas se permite la utilización de términos difusos en la condición de la consulta (predicados difusos que pueden ser unimodal o monótonos), obteniendo como resultado filas con grado de membresía de acuerdo a la función de membresía predefinida en cada caso. Por otro lado se hicieron consultas clásicas sobre la base de datos utilizando el RDBMS PostgreSQL versión 8.3, tomando en cuenta que estas consultas corresponden a las que derivan las consultas difusas efectuadas sobre PostgreSQLf y SQLfi.

Se pobló una base de datos PostgreSQL versión 8.3 utilizando el TPC Benchmark™H versión 2.9.0 [TPC11], generando tres volúmenes de datos diferentes: 1 GB, 5 GB y 10 Gb de datos; a estos volúmenes se les denominó bajo, alto y muy alto respectivamente. Se utilizó el mismo computador para todos los experimentos, Compaq® Presario con procesador Intel® core duo 1.83 GHz, 3 Gbyte en memoria RAM y disco duro de 160 Gbyte, sistema operativo Ubuntu 9.04. Además al tomar el tiempo de respuesta de cada consulta se inició el servidor de PostgreSQL para mantener las condiciones iniciales.

Para el volumen de datos muy alto el SQLfi no dio resultados, por lo que se demuestra que no es escalable. Es por ello que los experimentos de comparación se realizaron para el volumen bajo y alto.

Se crearon consultas variando diversos factores en dos niveles, a saber: Número de tablas (1 y 3), Número de predicados difusos unimodales (1 y 2), Número de predicados difusos monótonos (1 y 2). La combinación de cada uno de estos factores da como resultado ocho (8) consultas, las cuales se realizaron para cada volumen de datos (16 consultas), cada consulta se repitió una vez con los mismos factores (32 consultas), que luego se hicieron para cada tipo de acoplamiento (débil, fuerte y ninguno) por lo que se realizaron 96 experimentos en los cuales se midió el tiempo de respuesta (tiempo total de ejecución) y el número de filas resultado.

Para efectos de reproducibilidad del experimento se anexan los predicados difusos utilizados, las consultas difusas creadas fundamentadas en consultas del benchmark TPC-H y las consultas clásicas obtenidas aplicando el principio de derivación propuesto por Bosc y Pivert [BP00]. Además el software utilizado es de uso libre y fue distribuido en forma gratuita a través de actos públicos, además se encuentran disponibles en la siguiente dirección http://ldc.usb.ve/~jtcadenas/PostgreSQLf/.

Se ha demostrado en otros estudios con las implementaciones de débilmente y fuertemente acopladas que estos factores tienen influencia en los tiempos de respuesta [Cad08] [Tin06]. Mediante el número de filas resultado se comprueba funcionalmente que se está obteniendo el mismo resultado de cada una de las consultas (32) sobre los diferentes tipos de acoplamiento.

No se hacen experimentos con otros servidores de Bases de Datos distintos a PostgreSQL, ya que la implementación de la estrategia de acoplamiento fuerte esta desarrollada con este DBMS, y es de interés comparar el rendimiento de esta estrategia versus consultas clásicas. Asimismo, para este estudio no es de utilidad variar condiciones como el equipo de cómputo porque no estamos comparando la eficiencia de diversos DBMS lo cual puede ser consultado en revistas especializadas del área, sino la aplicabilidad de la implementación de SQLf y su efecto sobre el rendimiento.

En cuanto a otras implementaciones realizadas de consultas difusas destaca la de FSQL [Gal05] la cual también incorpora la representación de información incierta, pero la implementación no está basada en la teoría de SQLf, por lo que no puede hacerse una comparación. Una propuesta para la estandarización de los lenguajes utilizados en FSQL y SQLf es propuesta en [UTG08] y bajo la premisa de esta posible estandarización hacen una comparación de ambas implementaciones.

4. Resultados Experimentales

En la Tabla 1 se resumen estadísticos descriptivos obtenidos a través del paquete estadístico [SPSS06], se muestra una diferencia entre la media y la mediana de 4.9450 segundos, esta última no está influenciada por valores atípicos (*outliers*).

En cuanto a la dispersión, se tiene que la diferencia entre el valor mínimo y el máximo es alta (0.61 y 6553.94 segundos), además debido a que los dos primeros cuartiles están cerca (1.36 y 4.945 segundos) significa que un 25% de la muestra tiene tiempos de respuesta muy parecidos, por lo que existe una concentración en ese intervalo.

Se puede observar la dispersión en el diagrama de caja presentado en la Figura 1, se utilizó una escala logarítmica con base 10 para el tiempo, se tomó como etiqueta de valores atípicos el número de tablas; se nota que cuando existen 3 tablas se producen los valores atípicos. La mayor concentración de tiempos de respuesta de consultas está entre 0.61 y 4.9450, donde se ubican el 50% de las consultas, más aún hasta 18.6 segundos está el 75% de las mismas, observándose gran dispersión en el cuarto cuartil, con varios puntos *outliers* después de los 1000 segundos.

Tabla 1. *Estadísticos descriptivos*

N	Válidos	96
	Perdidos	0
Media		276.4136
Mediana		4.9450
Moda		.83(a)
Desv. típ.		1003.55607
Mínimo		.61
Máximo		6553.94
Percentiles	25	1.3600
	50	4.9450
	75	18.6000

(a) Existen varias modas. Se muestra el menor de los valores.

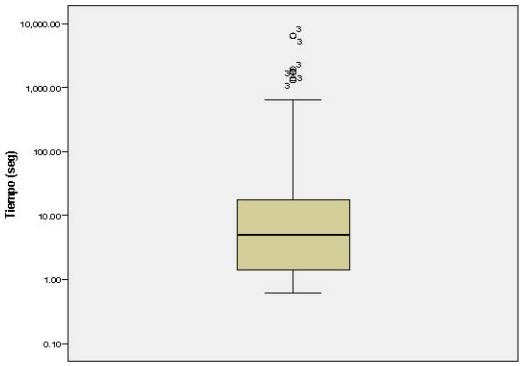

Figura 1. *Diagrama de Caja*

El resultado al utilizar como variable de agrupación el tipo de acoplamiento se visualiza en la Figura 2. Se observa que los mayores tiempos de respuesta se presentan con el tipo de acoplamiento débil, además estos son los que influyen en la variabilidad.

En cuanto a los tiempos de respuesta del acoplamiento fuerte y ninguno, están entre el mínimo y 20 segundos. El resumen de las medias agrupadas por tipo de acoplamiento puede observarse en la Tabla 2.

Se observa que para el caso de acoplamiento débil, tanto la media y la desviación típica son altas con respecto a los otros dos tipos de acoplamiento.

Para comprobar estadísticamente la diferencia significativa de las medias entre el tipo de acoplamiento débil y los otros dos, se realizó un test estadístico ANOVA (ver Tabla 3).

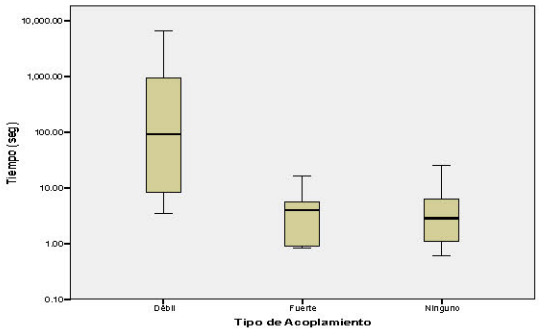

Figura 2: *Diagrama de Caja agrupado por Tipo de Acoplamiento*

Tabla 2. *Medias por tipo de Acoplamiento (tiempo en segundos)*

Tipo de Acoplamiento	Media	N	Desv. típ.
Débil	819.8222	32	1621.44022
Fuerte	4.8188	32	4.78318
Ninguno	4.6000	32	5.58951
Total	276.4136	96	1003.55607

Tabla 3. *Pruebas de los efectos inter-sujetos*

Fuente	Suma de cuadrados tipo III	Gl	Media cuadrática	F	Significación
Modelo corregido	14174056299282.950(a)	2	7087028149641.470	8.087	.001
Intersección	7334791846357.370	1	7334791846357.370	8.369	.005
Acoplamiento	14174056299282.980	2	7087028149641.490	8.087	.001
Error	81502785627573.000	93	876374039006.162		
Total	103011633773213.400	96			
Total corregida	95676841926856.000	95			

Variable dependiente: Tiempo (ms), R cuadrado = .148 (R cuadrado corregida = .130)

Se puede asegurar que existe por lo menos una diferencia entre las medias de las agrupaciones por tipo de acoplamiento, tal como se visualiza en la Figura 3, donde se nota la diferencia drástica entre las medias del acoplamiento fuerte y ninguno con respecto al tipo de acoplamiento débil.

Mediante un análisis post-hoc de Tukey, se reafirma lo observado, es decir, no se pueden establecer diferencias significativas estadísticamente con las muestras de datos entre los tipos de acoplamiento fuerte y ninguno, de acuerdo a lo mostrado en la Tabla 4.

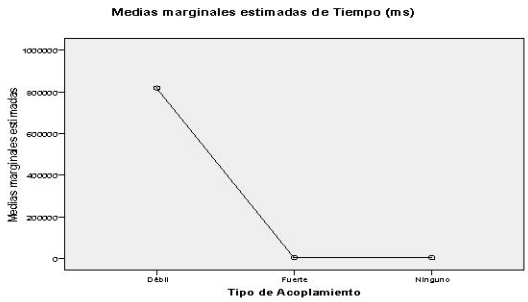

Figura 3: *Medias por tipo de Acoplamiento*

Tabla 4. *Subconjuntos Homogéneos (Tiempo ms)*

Tipo de Acoplamiento	N	Subconjunto	
	1	2	1
Ninguno	32	4598.66	
Fuerte	32	4818.58	
Débil	32		819821.41
Significación		1.000	1.000

5. Conclusiones y Trabajos Futuros

A través del análisis estadístico formal, se pudo comprobar que existe diferencia entre los tiempos de respuesta de los tipos de acoplamiento (fuerte, débil y ninguno) de los sistemas de consultas difusas, donde no hay diferencias significativas entre las medias muestrales de los acoplamientos de tipo fuerte y ninguno. Este es un resultado halagador para la implementación del tipo de acoplamiento fuerte, ya que indica que se están ofreciendo beneficios novedosos (consultas difusas) en sistemas gestores de bases de datos sin afectar significativamente el rendimiento.

Además se demostró que la implementación utilizando una estrategia de acoplamiento débil no escala para grandes volúmenes de datos.

En la estrategia de implementación de consultas difusas con acoplamiento fuerte sobre el PostgreSQL, el grado de membresía se calcula en el kernel del RDBMS. Mientras que en la estrategia de acoplamiento débil una vez derivada la consulta difusa a una consulta clásica, se obtiene el conjunto resultado y se calcula el grado de membresía, incurriendo en costos adicionales por reprocesamiento

Se evidencia la importancia de la utilización de estrategias de acoplamiento fuerte para obtener mejoras en el rendimiento del procesamiento de consultas difusas y la escalabilidad; permitiendo flexibilizar el lenguaje de consulta SQL y el manejo de preferencias del usuario.

Es importante analizar para futuros trabajos el impacto en el modelo de costos del optimizador cuando se implementa SQLf con una estrategia de acoplamiento fuerte.

Es posible continuar con la extensión del PostgreSQLf para soportar todos los operadores y funciones de SQLfi, incluyendo las extensiones para el estándar de SQL 2003.

Se recomienda estudiar la extensión de consultas difusas en RDBMS utilizando complementos (*plug-ins*) para que no se dependa exclusivamente de una versión tal como sucede en el caso del PostgreSQLf, mejorando la portabilidad entre versiones y medir su impacto en el rendimiento a través de nuevas pruebas experimentales.

Agradecimiento

Al FONACIT (Proyecto No. G-2005000278) por su apoyo para el desarrollo de esta investigación. *Dad gracias al Señor porque es bueno, porque es eterna su misericordia* (Salmo 117).

Referencias

[ACT11] Aguilera A., Cadenas J. T. y Tineo L.: Fuzzy Querying Capability at Core of a RDBMS. In Yan L., and Ma, Z.(Eds): *Advanced Database Query Systems: Techniques, Applications and Technologies*, 160 - 184. IGI Global New York, Estados Unidos (2011)

[AR11] Aguilera, A. y Rodríguez, R: Representación y manipulación de datos médicos en marcha patológica. *Multiciencias*, Vol. 11, No. 1, pp. 76-84, (2011).

[BS07] Berman, R. y Saunders, M.: *Dealing with Statistics. What you Need to Know*. Open University Press. Buckingham, GBR. McGraw-Hill Education (2007).

[BP95] Bosc, P. y Pivert, O.: SQLf: a relational database language for fuzzy querying. *Fuzzy Systems, IEEE Transactions on*, vol.3, no.1, pp.1-17 (1995)

[BP00] Bosc P. y Pivert O.: SQLf query functionality on top of a regular relational DBMS. In Pons, Vila, and Kacprzyk (Eds.): *Knowledge Management in Fuzzy Databases* (2000).

[BDM*05] Boulos J., Dalvi N., Mandhani B., Mathur S., Re C. y Suciu D.: MystiQ: A system for finding more answers by using probabilities. *System Demo in ACM SIGMOD International Conference on Management of Data* (2005).

[Cad08] Cadenas J.T: *Una Contribución a la Interrogación Flexible de Bases de Datos: Optimización y Evaluación a Nivel Físico.* Trabajo de Grado de Maestría presentado en la Universidad Simón Bolívar, Venezuela (2008).

[CB05] Connolly T. y Begg C.: *Database Systems - a Practical Approach to Design, Implementation and Management*, Pearson

Education Limited, United Kingdom (2005).

[Che09] Cheng, R.: Querying and Cleaning Uncertain Data. In Rothermel K; Fritsch D; Blochinger W; Durr F (Eds.) *1st International Workshop on Quality of Context*. LNCS, Vol. 5786, pp. 41-52. Stuttgart, Germany, (2009).

[DAC*08] Delgado, G., Aranda, V., Calero, J., Sánchez-Marañón, M., Serrano, J. M., Sánchez, D. & Vila, M. A. (2008). Building a fuzzy logic information network and a decision-support system for olive cultivation in Andalusia. *Spanish Journal of Agricultural Research*, Vol 6, No. 2, pp. 252 – 263.

[Gal05] J. Galindo, "New Characteristics in FSQL, a Fuzzy SQL for Fuzzy Databases". WSEAS Transactions on Information Science and Applications 2, Vol. 2, pp. 161-169 (2005).

[Gal08] Galindo, J. (Ed.): Handbook of Research on Fuzzy Information Processing in Databases. Hershey, PA, USA: Information Science (2008).

[GB09] Grissa Touzi, A. y Ben Hassine, M.A.: New Architecture of Fuzzy Database Management Systems. *The Int. Arab Journal of Inf. Tech.,*Vol 6, No. 3, pp. 213-220,

[GT01a] Goncalves M. y Tineo L.: SQLf Flexible Querying Language Extension by means of the norm SQL2, In *Proc of FUZZ-IEEE* (2001).

[GT01b] Goncalves M. y Tineo L.: SQLf3: An extension of SQLf with SQL3 features, In *Proc of FUZZ-IEEE* (2001).

[GT08] Goncalves M. Y Tineo L.: SQLfi and its Applications. *Avances en Sistemas e Informática*, Vol 5 No. 2. Medellin, ISSN 1657-7663 (2008).

[Koc09] Koch, C.: MayBMS: A Database Management System for Uncertain and Probabiilistic Data. In Aggarwal (Ed.), *Managing and Mining Uncertain Data* 149-184 (2009).

[LLL09] Le, V. H., Liu, F., y Lu, H.: A Data Model for Fuzzy Linguistic Databases with Flexible Querying. In Nicholson A; Li X (Eds.). *Proceedings of Advances in Artificial Intelligence*. Vol 5866, pp. 495 – 505. Springer_Verlag, (2009).

[LT06] López Y. y Tineo L.: About the Performance of SQLf Evaluation Mechanisms, *CLEI Electronic Journal, Volume 9, 2, Paper 8* (2006).

[MPV94] Medina M., Pons O., y Vila M.A.: An Elemental processor of Fuzzy SQL. *Computer Journal of Math Ware and Soft Computing*. Vol 1., No. 3, pp. 285-295, (1994).

[MY10] Ma, Z.M., y Yan, L.: A Literature Overview of Fuzzy Conceptual Data Modeling. In *Journal of Information Science and Engineering*. Vol. 26, No. 2, pp. 427-441 (2010).

[RCC09] Rodrígues, R.D., Cruz, A.J.O., y Cavalcante, R.T.: Aliança: A Proposal for a fuzzy databases architecture incorporating XML. *Fuzzy Sets and Systems*. Vol 160, pp. 269-279 (2009).

[SPSS06] SPSS: Versión 15.0 para Windows. Version 15.0.1. Copyrigth © SPSS Inc., (2006)

[Tim01] Timarán R.: Arquitecturas de Integración del Proceso de Descubrimiento de Conocimiento con Sistemas de Gestión de Bases de Datos: un Estado del Arte, *Ingeniería y Competitividad*, 3(2), (2001).

[Tin06] Tineo L.: A Contribution to Database Flexible Querying: Fuzzy Quantified Queries Evaluation. Disertación doctoral, Universidad Simón Bolívar, Venezuela (2006).

[TPC11] TPC: Transaction Processing Performance Council. http://www.tpc.org/ (2011)

[UGS10] Urrutia, A., Galindo, J., y Sepúlveda, A.: 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, (2010).

[UTG08] Urrutia A., Tineo L y Gonzalez C.: FSQL and SQLf: Towards a Standard in Fuzzy Databases. In Galindo (Ed.): *Handbook of Reasearch on Fuzzy Information Processing in Databases*, Volume 1, 270 – 298. Idea Group Inc. (2008).

[Wid09] Widom J. Trio: A system for Integrated Management of Data, Uncertainty, and Lineage. In Aggarwal (Ed.), *Managing and Mining Uncertain Data*, 113-148 (2009)

[Yag10] Yager, R.: Soft Querying of Standard and Uncertain Databases. *IEEE Transactions on Fuzzy Systems*. Vol 18, No. 2., (2010).

[Zad65] Zadeh L.A.: Fuzzy Sets, *Information and Control*, 8:338-353 (1965).

[Zad94] Zadeh L.: Soft Computing and Fuzzy Logic. *IEEE Software 11 (6)*, 48–56 (1994)

[ZM09] Zhao, F.X., y Ma, Z.M.: Vague Query Based on Vague Relational Model. In Yu W.; Sanchez (Eds.). *Advances in Computational Intelligence*. Vol 61, pp. 229 – 238, (2009).

Anexos

a) Predicados Difusos:

```
CREATE FUZZY PREDICATE cheap ON 0 .. 10000 AS (infinite, infinite, 100, 200);
CREATE FUZZY PREDICATE affordable ON 0 .. 10000 AS (500, 700, 900, 1000);
CREATE FUZZY PREDICATE expensive ON 0 .. 10000 AS (1000, 1500, infinite, infinite);
CREATE FUZZY PREDICATE small ON 0 .. 100 AS (infinite, infinite, 10, 15);
CREATE FUZZY PREDICATE big ON 0 .. 100 AS (40, 100, infinite, infinite);
CREATE FUZZY PREDICATE HIGHAVAIL ON 0 .. 10000 AS (2000, 4000, infinite, infinite);
CREATE FUZZY PREDICATE LOWAVAIL ON 0 .. 10000 AS (infinite, infinite, 1000, 1500);
```

b) Consultas Difusas:

1) select ps_partkey, ps_supplycost, ps_comment from partsupp where ps_availqty = LOWAVAIL AND ps_supplycost = affordable;

2) select ps_partkey, ps_supplycost, ps_comment from partsupp where (ps_supplycost = affordable OR ps_supplycost = cheap) AND ps_availqty = LOWAVAIL;

3) select ps_partkey, ps_supplycost, ps_comment from partsupp where ps_availqty = MEDIUMAVAIL AND (ps_supplycost = affordable OR ps_supplycost = cheap);

4) select ps_partkey, ps_supplycost, ps_availqty, ps_comment from partsupp where (ps_supplycost = affordable OR ps_supplycost = cheap) AND (ps_availqty = MEDIUMAVAIL OR ps_availqty = LOWAVAIL);

5) SELECT p_name, s_name FROM part, partsupp, supplier WHERE p_partkey = ps_partkey AND ps_suppkey = s_suppkey AND ps_availqty = LOWAVAIL AND ps_supplycost = affordable;

6) SELECT p_name, s_name FROM part, partsupp, supplier WHERE p_partkey = ps_partkey AND ps_suppkey = s_suppkey AND ps_availqty = LOWAVAIL AND ps_supplycost = affordable AND p_size = small;

7) SELECT p_name, s_name FROM part, partsupp, supplier WHERE p_partkey = ps_partkey AND ps_suppkey = s_suppkey AND ps_availqty = MEDIUMAVAIL AND ps_supplycost = affordable AND p_size = small;

8) SELECT p_name, s_name FROM part, partsupp, supplier WHERE p_partkey = ps_partkey AND ps_suppkey = s_suppkey AND ps_availqty = MEDIUMAVAIL AND ps_supplycost = affordable AND (p_size = medium OR p_size = small);

c) Consultas Clasicas:

```
1) SELECT ps_partkey, ps_supplycost, ps_comment
FROM partsupp
WHERE ((ps_availqty > 0.0) AND (ps_availqty < 1500.0)) AND ((ps_supplycost > 500.0) AND (ps_supplycost < 1000.0)) ;
2) SELECT ps_partkey, ps_supplycost, ps_comment
FROM partsupp
WHERE (((ps_supplycost > 500.0) AND (ps_supplycost < 1000.0)) OR ((ps_supplycost > 0.0) AND (ps_supplycost < 200.0))) AND ((ps_availqty > 0.0) AND (ps_availqty < 1500.0)) ;
3) SELECT ps_partkey, ps_supplycost, ps_comment
FROM partsupp
WHERE ((ps_availqty > 1500.0) AND (ps_availqty < 2000.0)) AND (((ps_supplycost > 500.0) AND (ps_supplycost < 1000.0)) OR ((ps_supplycost > 0.0) AND (ps_supplycost < 200.0))) ;
4) SELECT ps_partkey, ps_supplycost, ps_availqty, ps_comment
FROM partsupp
WHERE (((ps_supplycost > 500.0) AND (ps_supplycost < 1000.0)) OR ((ps_supplycost > 0.0) AND (ps_supplycost < 200.0))) AND (((ps_availqty > 1500.0) AND (ps_availqty < 2000.0)) OR ((ps_availqty > 0.0) AND (ps_availqty < 1500.0))) ;
5) SELECT p_name, s_name
FROM part , partsupp , supplier
WHERE p_partkey = ps_partkey AND ps_suppkey = s_suppkey AND ((ps_availqty > 0.0) AND (ps_availqty < 1500.0)) AND ((ps_supplycost > 500.0) AND (ps_supplycost < 1000.0)) ;
6) SELECT p_name, s_name
FROM part , partsupp , supplier
WHERE p_partkey = ps_partkey AND ps_suppkey = s_suppkey AND ((ps_availqty > 0.0) AND (ps_availqty < 1500.0)) AND ((ps_supplycost > 500.0) AND (ps_supplycost < 1000.0)) AND ((p_size > 0.0) AND (p_size < 15.0)) ;
7) SELECT p_name, s_name
FROM part , partsupp , supplier
WHERE p_partkey = ps_partkey AND ps_suppkey = s_suppkey AND ((ps_availqty > 1500.0) AND (ps_availqty < 2000.0)) AND ((ps_supplycost > 500.0) AND (ps_supplycost < 1000.0)) AND ((p_size > 0.0) AND (p_size < 15.0)) ;
8) SELECT p_name, s_name
FROM part , partsupp , supplier
WHERE p_partkey = ps_partkey AND ps_suppkey = s_suppkey AND ((ps_availqty > 1500.0) AND (ps_availqty < 2000.0)) AND ((ps_supplycost > 500.0) AND (ps_supplycost < 1000.0)) AND (((p_size > 15.0) AND (p_size < 40.0)) OR ((p_size > 0.0) AND (p_size < 15.0))) ;
```