Análisis de desempeño de fuzzydoDB*[1]

Performance Analysis of fuzzydoDB

Análise de Desempenho de fuzzydoDB

Soraya Odalis CARRASQUEL OROPEZA[2], David José CORONADO[3], Ricardo Rafael MONASCAL CASO[4], Carmen Rosseline RODRÍGUEZ DE TINEO[5], Leonid José TINEO RODRÍGUEZ[6]

Recibo: 01.08.2016 – Aprobación: 02.05.2017

Resumen: *En los últimos años se han propuesto extensiones a SQL para que permita atributos de datos difusos cuyo dominio es un conjunto de etiquetas provisto de una relación de similitud. Según Fuzzy EER, estos dominios se denominan Tipo 3. Dichas extensiones consideran la definición y manipulación de este tipo de dominios, así como consultas con ordenamiento y agrupamiento difuso. Mediante una arquitectura medianamente acoplada,*

* Modelo para la citación de este artículo / Template for citation of this article / Modelo para a citação deste artigo: CARRASQUEL OROPEZA, Soraya Odalis; CORONADO, David José; MONASCAL CASO, Ricardo Rafael; RODRÍGUEZ DE TINEO, Carmen Rosseline & TINEO RODRÍGUEZ, Leonid José (2017). Análisis de desempeño de fuzzydoDB. En: Ventana Informática No. 36 (ene-jun). Manizales (Colombia): Facultad de Ciencias e Ingeniería, Universidad de Manizales. p. 17-34. ISSN: 0123-9678

1 Artículo de investigación científica y tecnológica / Article of scientific and technological research / Artigo de pesquisa científica e tecnológica
Proyecto / Project / Projeto: Desafíos del Modelo Relacional Difuso / Challenges of the Diffuse Relational Model / Desafios do Modelo Relacional Difuso
Periodo / Period / Período: enero 2013 - en progreso / January 2013 - in progress / Janeiro 2013 – em andamento
Institución / Institution / Instituição: Universidad Nacional Experimental Politécnica "Antonio José de Sucre" (Barquisimeto, Lara, Venezuela).

2 Magister en Matemáticas / Master in mathematics / Mestre em Matemática. Profesor Agregado / Added Professor / Profesor Agregado, Universidad Simón Bolívar (Caracas, Miranda, Venezuela). scarrasquel@usb.ve. http://orcid.org/0000-0003-1313-1387

3 Doctor en Matemáticas / Doctor en Mathematics / Doutor em Matemática. Profesor Agregado / Added Professor / Profesor Agregado, Universidad Simón Bolívar (Caracas, Miranda, Venezuela). Correo electrónico: dcoronado@usb.ve. http://orcid.org/0000-0002-4455-1184

4 Magister en Ciencias de la Computación / Master in Computing Science / Mestre em Ciência da Computação. Profesor Asistente / Assistant Professor / Profesor Assistente, Universidad Simón Bolívar (Caracas, Miranda, Venezuela). rmonascal@usb.ve. http://orcid.org/0000-0002-7341-9166

5 Magister en Ciencias de la Computación / Master in Computing Science / Mestre em Ciência da Computação. Profesor Asociado / Associate Professor / Profesor Associado, Universidad Simón Bolívar (Caracas, Miranda, Venezuela). Profesor Invitado / Guest Lecturer / Profesor Visitante, Universidad Católica "Nuestra Señora de la Asunción" (Asunción, Paraguay). crodrig@usb.ve. http://orcid.org/0000-0002-8174-2932

6 Doctor en Computación / Doctor in Computing Science / Doutor em Computação. Profesor Titular / Titular Professor / Profesor Titular, Universidad Simón Bolívar (Caracas, Miranda, Venezuela). Profesor Invitado, Universidad Católica "Nuestra Señora de la Asunción" (Asunción, Paraguay). leonid@usb.ve. http://orcid.org/0000-0001-9677-4956

estas nuevas funcionalidades fueron añadidas a PostgreSQL, surgiendo así fuzzydoDB. En este trabajo se presenta el análisis de desempeño de las consultas con ordenamiento y agrupamiento basado en atributos difusos Tipo 3, así como una base de datos experimental para este tipo de estudios.

Palabras clave: *SGBD; ORDER BY; GROUP BY; atributos difusos Tipo 3; análisis de desempeño*

Abstract: *In recent years there have been proposed extensions to SQL to allow fuzzy data attributes whose domain is a set of labels provided with a similarity relation. According Fuzzy EER, these domains are called Type 3. These extensions consider the definition and manipulation of such domains as well as queries with fuzzy ordering and partitioning. Through a middle coupling architecture, these new features were added to PostgreSQL, thus resulting fuzzydoDB. In this paper we analyze query performance with ordering and grouping based on Type 3 fuzzy attributes, and an experimental database for such studies is presented.*

Keywords*: DBMS; ORDER BY; GROUP BY; Type 3 fuzzy attributes; performance analysis.*

Resumo: *Nos últimos anos tem havido extensões propostas para o SQL para permitir atributos de dados distorcido cujo domínio é um orcid.org/0000-0003-1313-1387Tipo 3. Estas extensões considerar a definição e manipulação de tais domínios, bem como consultas com ordenação e agrupamento difuso. Através de uma arquitetura moderadamente acoplado, estes novos recursos foram adicionados ao PostgreSQL, resultando assim fuzzydoDB. Neste artigo analisamos o desempenho da consulta com ordenação e agrupamento com base em atributos difusa Tipo 3, e um banco de dados experimental para tais estudos é apresentado*

Palavras-chave*: DBMS; ORDER BY; GROUP BY; atributos difusa do tipo 3; análise de desempenho*

Introducción

La mayoría de los gestores de bases de datos (SGBD) implementan el modelo relacional, que supone datos precisos o ausentes. Sin embargo, en el mundo real, puede haber información parcial o imprecisa sobre valores de algunos datos. Una herramienta matemática para modelar este tipo de información son los conjuntos difusos, concebida por Zadeh (2015, 6). La aplicación de esta teoría ha dado origen al

modelo relacional difuso generalizado GEFRED (Medina, Pons & Vila, 1994, 89), así como el modelo entidad relación extendido difuso Fuzzy EER (Galindo, Urrutia & Piattini, 2006, 14) y la extensión a SQL con conjuntos difusos FSQL (Galindo, 2008). Fuzzy EER distingue cuatro tipos de atributos difusos: *Tipo 1*, datos precisos sobre los que se permite consultas con etiquetas lingüísticas interpretadas como números difusos; *Tipo 2*, datos posibilísticos sobre dominios ordenados; *Tipo 3*, etiquetas lingüísticas no ordenadas provistas de una relación de similitud, y distribuciones de posibilidad sobre ellas; y *Tipo 4*, similar a los *Tipo 3* pero sin la relación de similitud.

Carrasquel, Rodríguez & Tineo (2013, 37-39; 2014, 522-524) han estudiado las implicaciones de atributos *Tipo 3* en consultas basadas en ordenamiento (*ORDER BY*) y agrupamiento (*GROUP BY*) sobre conjuntos de etiquetas con una relación de similitud, sin considerar las distribuciones de posibilidad. Los resultados teóricos obtenidos han sido implementados en una extensión del SGBD MariaDB, planteada por Carrasquel et al. (2014, 94-96), que permite especificar y almacenar atributos *Tipo 3* y procesar consultas que involucren estos atributos, particularmente, en *ORDER BY* y *GROUP BY* basado en relaciones de similitud, lo cual aumenta la funcionalidad del SGBD al proveer un lenguaje de consulta con mayor expresividad. Esta extensión fue migrada a *PostgreSQL* y ampliada para dar soporte a otros tipos de datos difusos, la cual se denominó *fuzzydoDB*, siguiendo a Coronado et al. (2015, 328).

Se planteó realizar el análisis de desempeño de *fuzzydoDB* en el procesamiento de consultas con agrupamiento y ordenamiento difuso sobre atributos *Tipo 3*. Para ello fue necesario crear una base de datos experimental que incluyera atributos *Tipo 3*, pues, dado lo reciente de esta extensión, no existen bases de datos públicas que contemplen estos atributos. Aquí se presenta dicha base de datos experimental y se reporta el resultado del análisis de desempeño.

Este artículo ha sido estructurado así: en la sección 1, el marco teórico constituido por los dominios difusos *basados en relaciones de similitud, así como algunos* aspectos relevantes de la extensión a *PostgreSQL*, denominada *fuzzydoDB*. En la sección 2 se explica el trabajo realizado de acuerdo a la metodología de experimentación, primero con la elaboración de la base de datos y luego el diseño experimental. En la Sección 3 se presentan los resultados obtenidos y se hace la correspondiente discusión. Finalmente, en la sección 4, se proponen las conclusiones y trabajos futuros.

1. Fundamento teórico

1.1 Relaciones de similitud

Un conjunto difuso *F* en un universo *F* se caracteriza por una función de membresía $\mu_F: X \rightarrow [0,1]$, según Zadeh (1965, 339). Cuánto más se acerca a 1 la membresía de un elemento, está más posiblemente (o certeramente) incluido en el conjunto. Así 0 es la medida de completa exclusión y 1 la de completa inclusión. Zadeh (1971, 177) propuso el concepto de relación de similitud, una extensión difusa de las relaciones de equivalencia, en tanto Carrasquel, Rodríguez & Tineo (2013, 31) definen una relación de similitud *S* como un conjunto difuso caracterizado por $\mu_S: X \times X \rightarrow [0,1]$, que es reflexiva $\forall x \in X \ \mu_S(x,x) = 1$, simétrica $\forall x,y \in X \ \mu_S(x,y) = \mu_S(y,x)$ y transitiva $\forall x,y,z \in X \ (\mu_S(x,y)=1 \wedge \mu_S(y,z)=\beta) \Rightarrow \mu_S(x,z)=\beta) \wedge \forall x,y,z \in X \ (\mu_S(x,y)=\beta \wedge \mu_S(y,z)=1) \Rightarrow \mu_S(x,z)=\beta)$. La Tabla 1(a) muestra un ejemplo.

La relación de similitud induce una partición difusa, indica Zadeh (1971, 187): Sea *S* una relación de similitud en *X*, *S* induce una partición difusa $P_S = \{ S[x] \mid x \hat{I} X \}$; cada *S*[*x*] es la clase difusa de *x*, conjunto difuso caracterizado por $m_{S[x]}(y) = m_S(x,y)$. La Tabla 1(b) muestra un ejemplo, en el cual se observa que las clases difusas de los elementos "Rojo" y "Vino" son la misma. Carrasquel, Rodríguez & Tineo (2013, 2014) proponen y formalizan extensiones al *SQL* para la definición de atributos *Tipo 3* y su uso en consultas ordenadas y agrupadas, que se resumen a continuación.

Tabla 1. (a) Relación de Similitud para Colores (b) Partición difusa para Colores

μ_s	Negro	Marrón	Morado	Rojo	Rosa	Vino
Negro	1	0.5	0	0	0	0
Marrón	0,5	1	0.3	0.2	0	0.2
Morado	0	0.3	1	0.8	0	0.8
Rojo	0	0.2	0.8	1	0.7	1
Rosa	0	0	0	0.7	1	0.7
Vino	0	0.2	0.8	1	0.7	1

(a)

x	clase difusa de x
Negro	{Negro/1, Marron/0.5}
Marrón	{Marron/1, Negro/0.5, Morado/0.3, Rojo/0.2, Vino/0.2}
Morado	{Morado/1, Marron/0.3, Rojo/0.8, Vino/0.8}
Rojo	{Rojo/1, Vino/1, Marron/0.2, Morado/0.8, Rosa/0.7}
Rosa	{Rosa/1, Rojo/0.7, Vino/0.7}
Vino	{Rojo/1, Vino/1, Marron/0.2, Morado/0.8, Rosa/0.7}

(b)

1.1.1 Dominios de datos con Relaciones de Similitud. Un dominio de atributos *Tipo 3* se especifica así: CREATE FUZZY DOMAIN *fd* AS VALUES l_1,…, l_n [SIMILARITY { (l_{i1}, l_{j1}) / v_1,…, (l_{im}, l_{jm}) / v_m }], donde *fd* es el nombre del dominio; l_1,…, l_n es la lista de etiquetas que definen el dominio; los especificadores (l_{ik},l_{jk})/v_k son los pares de la relación difusa de similitud para ese dominio siendo v_k el valor del grado de membresía de dicho par en la relación.

La relación de similitud es opcional, solo es necesario especificar los pares básicos de la relación, o relación base, pues los correspondientes a la reflexividad, simetría y transitividad están sobreentendidos. Los pares no especificados, que tampoco se puedan obtener mediante estas propiedades, tienen grado de membresía cero. Por ejemplo, si se quiere crear un dominio difuso para colores con la relación de similitud de la Tabla 1(a), se especifica con la sentencia: CREATE FUZZY DOMAIN colores AS VALUES (Negro, Marrón, Morado, Rojo, Rosa, Vino) SIMILARITY {(Negro, Marrón)/0.5, (Marrón, Morado)/0.3, (Marrón, Rojo)/0.2, (Morado, Rojo)/0.8, (Rojo, Rosa)/0.7, (Vino, Rojo)/1}.

1.1.2 Ordenamiento usando Similitud. La cláusula ORDER BY se extiende así: ORDER BY *criterio*$_1$,...,*criterio*$_o$. Cada *criterio*$_i$ es de la forma k_i d_i con k_i un atributo y d_i un especificador de orden ASC o DESC o START v_i. En este último caso k_i es un atributo *Tipo 3* y v_i una etiqueta en el dominio de k_i, las tuplas serán ordenadas descendentemente por el grado de membresía $m_{S[}v_{i]}$ asociado a la relación de similitud *S* del dominio de k_i. Hay que recordar que la cláusula ORDER BY no tiene el efecto de filtro, por lo que no descarta las etiquetas para las cuales $m_{S[}v_{i]}$ es 0, simplemente las coloca al final.

Por ejemplo, dada cierta tabla POKEMON(nombre, region, *color*) de siete filas, con *color Tipo 3* del dominio *colores*, la consulta SELECT nombre, region, *color* FROM POKEMON ORDER BY *color* START Morado; produce la Tabla 2(a) de tuplas ordenadas decrecientemente por la similitud a Morado según la Tabla 1(a).

1.1.3 Agrupamiento usando Similitud. Se extiende la cláusula GROUP BY así: GROUP BY [SIMILAR] k_1, …, [SIMILAR] k_o, la palabra clave SIMILAR es opcional, sólo se usa si k_i es *Tipo 3* y se quiere considerar la partición difusa inducida por su relación de similitud. Por ejemplo: dada la tabla POKEMON de la sección anterior, la consulta SELECT region, color, COUNT(*) FROM POKEMON GROUP BY region, SIMILAR color; produce la Tabla 2(b) que usa la partición difusa de la Tabla 1(b). La función de agregación COUNT se calculó según el operador ∑*count* de Zadeh (1983, 154).

Tabla 2. Resultado de la consulta con (a) ordenamiento difuso (b) agrupamiento difuso

nombre	region	color
Crobat	Johto	Morado
Charizard	Kanto	Rojo
Wurmple	Hoenn	Rojo
Pidgey	Kanto	Marron
Sentret	Johto	Marron
Skitty	Hoenn	Rosa
Cherrim	Sinnoh	Rosa

(a)

region	color	COUNT
Hoenn	Rojo	1.7
Hoenn	Rosa	1.7
Johto	Marrón	1.3
Johto	Morado	1.3
Kanto	Marrón	1.2
Kanto	Rojo	1.2
Sinnoh	Rosa	1.0

(b)

1.2 SGBD difuso *fuzzy*doDB

Aquí se describe brevemente el SGBD *fuzzydoDB*, su arquitectura y catálogo (Carrasquel et al, 2014). La versión actual de *fuzzydoDB* (Coronado et al, 2015) incluye todas las variantes de atributos difusos según se propone en Fuzzy EER (Galindo, Urrutia & Piattini, 2006). Sin embargo, para *fuzzydoDB* se ha preferido extender la clasificación a *Tipo 2*, *Tipo 3*, *Tipo 4* y *Tipo 5*, donde los atributos *Tipo 3* no permiten el uso de distribuciones de posibilidad, mientras que los *Tipo 5* son la variante que sí las permite. A pesar que *fuzzydoDB* provee esta variedad de atributos difusos, solo se describe la porción correspondiente a los *Tipo 3*.

1.2.1 Catálogo Difuso. Los metadatos correspondientes a dominios *Tipo 3*, se incluyen en el catálogo de la base de datos mediante esquemas relacionales descritos a continuación. Las claves primarias se destacan con subrayado simple, las alternas con subrayado doble y las foráneas con el nombre de la tabla referenciada como un superíndice.

DOMAIN(domainId, tableSchema, domainName) contiene los diferentes dominios de atributos *Tipo 3*, domainId es autogenerada para facilitar su manipulación. LABEL(labelId, domainId^{DOMAIN}, labelName) almacena las etiquetas (labelName) de los dominios difusos, referenciados por domainId y labelId es autogenerada. SIMILARITY(label1Id^{LABEL}, label2Id^{LABEL}, value, derived) contiene los pares de etiquetas de las relaciones de similitud, con su grado de membresía (value) y derived indica si el par es o no derivado por reflexividad, simetría y transitividad. COLUMN(tableSchema, tableName, columnName, domainId^{DOMAIN}) contiene la definición de columnas cuyo tipo es un dominio difuso (domainId).

1.2.2 Arquitectura. Para extender la funcionalidad de un SGBD se han propuesto tres arquitecturas de acoplamiento: fuerte, débil y medio (Timarán, 2001). *fuzzydoDB* usa acoplamiento medio (Figura 1), que consiste en una implementación intermedia donde la lógica de la extensión[7] fue programada en el lenguaje nativo del SGBD original.

En el análisis sintáctico se identifican las columnas correspondientes a atributos *Tipo 3*, pues requieren de un trato especial. Para ello, se recorren las expresiones en las cláusulas SELECT, WHERE, ORDER BY y GROUP BY. Se asocia cada columna con su tabla y se busca en el catálogo difuso si ésta almacena datos de *Tipo 3* generando la lista de columnas a ser traducidas. Si la columna aparece en varias sub-expresiones de la consulta, se inserta una sola vez.

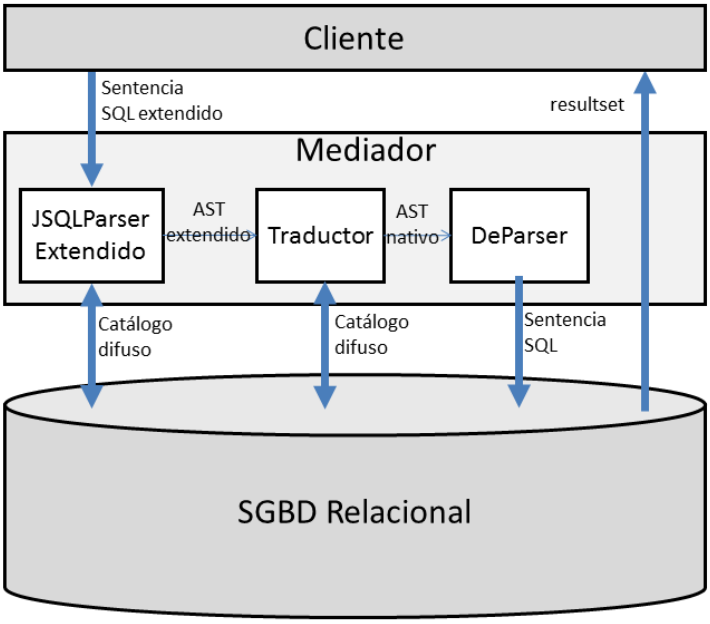

Figura 1. Arquitectura de fuzzydoDB (Carrasquel et al., 2014)

Luego de ser validada la sentencia original, se invoca al traductor, que utiliza el catálogo para generar un AST correspondiente a la sentencia traducida al lenguaje nativo de SQL, de acuerdo a los esquemas de

7 Esta extensión posee una capa externa programada en Java, un mediador que consiste fundamentalmente de tres módulos: *front-end*, *core* y *back-end*. El *front-end* es un analizador sintáctico o *parser* de SQL extendido que genera un árbol sintáctico abstracto (AST). El *core* es un traductor que transforma un AST del lenguaje extendido a un AST del lenguaje nativo. El *back-end* (*deparser*) es un generador de código que toma el AST y escribe la instrucción en SQL.

traducción (Carrasquel et al, 2016). Posteriormente se recorre de nuevo el AST para localizar expresiones que involucren columnas con datos difusos. Estas columnas son sustituidas en el AST por la columna de la tabla LABEL que contiene la etiqueta conocida por el usuario. Si esta sustitución se realizó en la cláusula SELECT, se asegura que la expresión mantenga su alias o el nombre por el cual se accede al valor en el resultado.

La generación de código realiza el proceso inverso para transformar el AST obtenido en un *string* SQL, esto es lo que se conoce como un *deparser*. Finalmente, se envía el *string* con la instrucción SQL traducida al manejador para que sea ejecutada, obteniendo el resultado de la instrucción original.

2. Metodología

2.1 Base de datos experimental

Se creó una base de datos experimental, que contiene personajes de Pokémon, una serie de videojuegos sobre pequeños monstruos de bolsillo que luchan batallas, ellos pueden ser capturados y entrenados. Los Pokémon tienen características idóneas para ser representadas mediante atributos difusos. Dado que esta serie de videojuegos es muy popular, puede ayudar a llamar la atención de personas interesadas en los dominios difusos. La base de datos se pobló con información tomada de la página oficial de The Pokemon Company (2011) y el *fan site* en español (Wikia, 2014).

2.1.1 Modelo lógico. La base de datos está compuesta de tres relaciones. La principal, llamada POKEMON, poblada con los 721 *Pokémon* existentes hasta ahora (Wikia, 2014). Las otras dos se llaman HABILIDADES y MEGAPOKEMON. El modelo lógico de la base de datos es el siguiente:

- POKEMON (codigo, nombre, *tElem1*, *tElem2*, *gHuevo1*, *gHuevo2*, *poder*, region, *ratio*, *color*, habilidad1^{HABILIDADES}, habilidad2^{HABILIDADES}, habilidad3^{HABILIDADES}). Un *Pokémon* es identificado por un código de tres dígitos en el rango '001'...'721' que representa al *Pokémon* según el orden del videojuego. Un *Pokémon* puede tener uno o dos tipos elementales (*tElem1* y *tElem2*), así como, uno o dos grupos huevo (*gHuevo1* y *gHuevo2*). Un *Pokémon* tiene un nivel de *poder* base estimado, una región donde es introducido por primera vez cuyos

posibles valores son ‘Kanto’, ’Johto’, ’Hoenn’, ’Sinnoh’, ‘Teselia’, ‘Kalos’, la *ratio* de captura y su *color*. Además, un Pokémon puede tener hasta tres habilidades. Esta relación tiene dos atributos difusos *Tipo 2* (*poder* y *ratio*) y cinco atributos difusos *Tipo 3* (*tElem1*, *tElem2*, *gHuevo1*, *gHuevo2* y *color*) cuyos dominios se explican más adelante.

- HABILIDADES (nombre, efecto) contiene las 192 habilidades posibles que puede tener un *Pokémon*, caracterizadas por su nombre y el efecto que causa.
- MEGAPOKEMON (codigo, nombre, *tElem1*, *tElem2*, *poder*, region, color, habilidad^{HABILIDADES}) almacena los 48 *Pokémon* de categoría mega que existen, que tienen los mismos atributos que POKEMON, salvo los grupos huevo y ratio.

2.1.2 Dominios difusos. Para modelar las características difusas de los *Pokémon* se definieron cinco dominios: dos *Tipo 2* (PODERBASE y RATIOCAPTURA) y tres son *Tipo 3* (TIPOELEMENTAL, GRUPOHUEVO y COLORES). Para estos últimos, se muestra la definición en SQL extendido pues son el objeto de este trabajo:

- TIPOELEMENTAL: existen 18 tipos elementales de *Pokémon,* con cierta similitud entre ellos, por lo que puede modelarse como un atributo difuso *Tipo 3*. La definición del dominio difuso en SQL extendido es CREATE FUZZY DOMAIN TIPOELEMENTAL AS VALUES (planta, fuego, agua, electrico, bicho, normal, volador, lucha, veneno, tierra, roca, acero, dragon, fantasma, hielo, psiquico, siniestro, hada) SIMILARITY {(planta, fuego)/0.1, (planta, agua)/0.5, (planta, bicho)/0.7, (planta, veneno)/0.2, (planta, tierra)/0.9, (fuego, electrico)/0.6, (fuego, acero)/0.3, (fuego, dragon)/0.8, (fuego, hielo)/0.2, (agua, electrico)/0.8, (agua,tierra)/0.1, (agua, roca)/0.3, (agua, hielo)/0.9, (electrico, dragon)/0.3, (bicho, volador)/0.6, (bicho, veneno)/0.4, (bicho, tierra)/0.3, (bicho, psiquico)/0.2, (normal, lucha)/0.4, (volador, dragon)/0.9, (volador, fantasma)/0.7, (volador, psiquico)/0.2, (volador, hada)/0.8, (lucha, roca)/0.5, (lucha, acero)/0.6, (lucha, psiquico)/0.3, (veneno, siniestro)/0.7, (tierra, roca)/0.9, (tierra, acero)/0.1, (acero, psiquico)/0.1, (dragon, siniestro)/0.3, (dragon, hada)/0.5, (fantasma, psiquico)/0.5, (fantasma, siniestro)/0.9, (psiquico, siniestro)/0.3 };
- GRUPOHUEVO: los *Pokémon* pueden aparearse para tener una cría, con la restricción que la pareja debe pertenecer al mismo grupo huevo. Los valores de este grupo son etiquetas que tienen cierta similitud por lo que se definió como un dominio *Tipo 3*, donde resalta que uno de los pares de la relación de similitud (Ninguno y Ditto)

permite la transitividad. La sentencia para definir este dominio es CREATE FUZZY DOMAIN GRUPOHUEVO AS VALUES (Ninguno, Ditto, Planta, Bicho, Volador, Humanoide, Amorfo, Mineral, Campo, Agua 1, Agua 2, Agua 3, Monstruo, Hada, Dragon) SIMILARITY { (Ninguno, Ditto)/1, (Planta, Bicho)/0.7, (Planta, Mineral)/0.5, (Planta, Campo)/0.8, (Bicho, Volador)/0.6, (Bicho, Campo)/0.7, (Volador, Hada)/0.8, (Volador, Dragon)/0.9, (Amorfo, Mineral)/0.3, (Amorfo, Monstruo)/0.8, (Mineral, Campo)/0.6, (Agua 1, Agua 2)/0.9, (Agua 1, Agua 3)/0.8, (Agua 2, Agua 3)/0.9, (Monstruo, Dragon)/0.6, (Hada, Dragon)/0.5};

- COLORES: los *Pokémon* pueden ser clasificados según el color al que pertenecen. Existen 10 colores, posibles de modelar según su similitud como un atributo *Tipo 3*. Dado que los videojuegos de la serie son en colores, se considera blanco, negro y gris como miembros de una misma clase[8]. La sentencia para definir este dominio es CREATE FUZZY DOMAIN COLORES AS VALUES (Azul, Amarillo, Blanco, Gris, Marron, Morado, Negro, Rojo, Rosa, Verde) SIMILARITY { (Azul, Morado)/0.8, (Azul, Verde)/0.8, (Amarillo, Blanco)/0.2, (Amarillo, Verde)/0.8, (Blanco, Gris)/1, (Blanco, Rosa)/0.7, (Gris, Negro)/1, (Marron, Morado)/0.3, (Marron, Negro)/0.5, (Morado, Rojo)/0.8, (Rojo, Rosa)/0.7 };

2.2 Diseño experimental

Jain (1991, 274-318) presenta el método de estudio experimental de desempeño usando un modelo estadístico formal, descrito por Castejón (2011, 49-68), que permite hacer el análisis de desempeño con una cantidad reducida de corridas, garantizado por el uso de estadística formal. La idea del método es explicar la influencia de diversos factores considerados en los valores observados de los experimentos. La importancia de un factor es medida por la proporción del total de variación en la respuesta que es explicada por el factor.

Más que estudiar valores absolutos de los resultados, se observan tendencias medias que permiten predecir el comportamiento esperado con la escalada de los niveles de los factores involucrados, de forma que un pequeño experimento puede arrojar conclusiones significativas, aunque los niveles de los factores no sean llevados a grandes escalas. Para el estudio se adoptó un diseño factorial 2^k, donde k es el número de factores utilizados, considerando únicamente dos niveles por factor,

8 No tiene que ser así… por razones ilustrativas de la transitividad y sus efectos, se decidió tomarlo de esta manera.

con cuatro réplicas por cada prueba para obtener mayor precisión, de manera que para cada experimento se realizan $2^{(k+2)}$ corridas[9].

Se tuvieron dos grupos de experimentos: el primero considera el impacto del tipo de atributo en *fuzzydoDB* y el segundo, el impacto de *fuzzydoDB* con atributos precisos respecto al SGBD original. En cada grupo hay dos experimentos, según el tipo de consulta: ORDER BY (E_1 y E_3) y GROUP BY (E_2 y E_4). En el primer grupo de experimentos (E_1 y E_2) se consideran los factores: volumen (*V*) y tipo de atributo (*A*). El factor *A* se refiere al atributo involucrado en la cláusula que se está experimentando, el cual puede ser preciso (A_1) o difuso *Tipo 3* (A_2). Para los experimentos del segundo grupo (E_3 y E_4) se consideran los factores: volumen (*V*) y SGBD (*S*). El factor S se refiere al SGBD sobre el cual se hace la consulta, que puede ser *PostgreSQL* original (S_1) o *fuzzydoDB* (S_2). En todos los experimentos se consideró el factor *V* con sus dos niveles: bajo (V_1) que corresponde a la tabla POKEMON con sus 721 registros (271.8 KB); y alto (V_2) que consiste de 72.100 filas (27,3 MB) obtenidas repitiendo 100 veces cada registro original.

Se diseñaron cuatro consultas experimentales. Las consultas C_1 y C_2 se usan en el experimento E_1, C_3 y C_4 en E_2, C_1 en E_3 y C_2 en E_4. Se decidió no usar cláusula WHERE a fin de concentrar la atención en el impacto del uso del atributo difuso en las cláusulas ORDER BY y GROUP BY. Las consultas son las siguientes:

C_1: SELECT nombre FROM POKEMON ORDER BY nombre;

C_2: SELECT *color* FROM POKEMON ORDER BY *color* START 'Amarillo';

C_3: SELECT region, count(*) AS total FROM POKEMON GROUP BY region;

C_4: SELECT *color*, count(*) AS total FROM POKEMON GROUP BY SIMILAR *color*;

Para la corrida de los experimentos, se fijaron las características del hardware, carga del computador y conexiones. Se usó una máquina VIT P2400 con procesador Intel Core i3-3110M a 2.4 GHz, RAM de 1.8GB y HDD de 150 GB. El disco estaba particionado para alojar dos sistemas de operación y las pruebas se realizaron en Ubuntu 14.04 LTS de 32 bits. En cuanto a la carga de la máquina, se tenía únicamente como proceso operativo el SGBD. Las conexiones a

9 Este tipo de diseño es muy utilizado para el análisis de desempeño de sistemas en que alguno de los factores pudiera tener escalas muy variadas. El número de réplicas aumenta los grados de libertad al momento de ajustar el modelo estadístico.

las bases de datos trabajaron sobre el servidor local de la máquina y para garantizar igualdad de condiciones entre las corridas de los experimentos, antes de ejecutar cada consulta, se limpiaba la memoria caché del SGBD.

La variable observada en el estudio fue el tiempo de ejecución (*T*) de las consultas medido en milisegundos, obtenido con el comando *time*. El diseño experimental se corresponde con el modelo aditivo $T=T'+\beta_1 A+\beta_2 V+\beta_3 AV+\varepsilon$, para el primer grupo de experimentos, y $T=T'+\beta_1 S+\beta_2 V+\beta_3 SV+\varepsilon$, para el segundo grupo.

3. Resultados y discusión

3.1 Presentación de resultados

Los resultados de los experimentos son presentados en forma numérica en la Tabla 3, los cuales se cargaron en el software estadístico R (R Core Team, 2015), para hacer las pruebas de ajuste del modelo aditivo, cuyos resultados se compilan en la Tabla 4. Finalmente, los resultados también se presentan en gráficos de interacción de factores (Figura 2).

Tabla 3. Tiempo (T) en ms observado en las corridas de las distintas réplicas (R) de los experimentos.

Primer grupo de experimentos: según tipo de Atributo					Segundo grupo de experimentos: según SGBD				
A	V	R	E_1	E_2	S	V	R	E_3	E_4
A_1	V_1	R_1	3954	4098	S_1	V_1	R_1	3716	3597
		R_2	4101	4081			R_2	3538	2922
		R_3	4352	4191			R_3	3782	2860
		R_4	4038	4015			R_4	3369	3718
	V_2	R_1	8815	5436		V_2	R_1	6360	2726
		R_2	9231	4187			R_2	6398	2938
		R_3	8762	5495			R_3	6682	2887
		R_4	8200	4534			R_4	5982	2849
A_2	V_1	R_1	4337	4668	S_2	V_1	R_1	3954	4098
		R_2	4323	4147			R_2	4101	4081
		R_3	4465	4208			R_3	4352	4191
		R_4	4464	4188			R_4	4038	4015
	V_2	R_1	6998	4494		V_2	R_1	8815	5436
		R_2	6967	4400			R_2	9231	4187
		R_3	8028	6184			R_3	8762	5495
		R_4	6887	5255			R_4	8200	4534

Tabla 4. Análisis de Varianza (ANOVA) de los resultados obtenidos en los experimentos

Experimento		Df	Sum Sq	Mean Sq	F value	Pr(>F)	
E_1	A	1	1552516	1552516	12.24	0.004389	**
	V	1	55703832	55703832	439.30	8.06e-11	***
	A:V	1	3305124	3305124	26.07	0.000259	***
	Residuals	12	1521607	126801			
E_2	A	1	141941	141941	0.482	0.500700	
	V	1	2551208	2551208	8.663	0.012300	*
	A:V	1	1314	1314	0.004	0.947800	
	Residuals	12	3533790	294483			
E_3	S	1	8447742	8447742	103.64	2.95e-07	***
	V	1	54686025	54686025	670.94	6.69e-12	***
	S:V	1	3558882	3558882	43.66	2.51e-05	***
	Residuals	12	978082	81507			
E_4	S	1	8323225	8323225	52.019	1.07e-05	***
	V	1	154056	154056	0.963	0.34584	
	S:V	1	1540081	1540081	9.625	0.00915	**
	Residuals	12	1920040	160003			
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1							

3.2 Discusión de resultados

3.2.1 Primer grupo de experimentos: Según Tipo de Atributo. Pretende evidenciar la influencia del tipo de atributo (preciso o difuso) en el desempeño de *fuzzydoDB*.

Consultas ORDER BY. La ANOVA de la Tabla 4, experimento E_1, indica que tanto el factor volumen como su interacción con el tipo de atributo, influyen de manera significativa en el tiempo de ejecución de las consultas con ORDER BY, mientras que la influencia del factor tipo de atributo es menos significativa. La Figura 2a señala que el crecimiento del tiempo de ejecución aumenta según el volumen de datos donde la tendencia es más fuerte para las consultas con atributos precisos, lo cual puede explicarse porque en el caso de las consultas con atributos precisos, las *tuplas* se ordenaron con base en cadenas de caracteres correspondientes a los valores de la columna *nombre* de la tabla POKEMON, mientras que en el caso de atributos difusos se ordenan en base al grado de similitud de la etiqueta para el atributo *color* respeto al valor *Amarillo*, dados en números reales[10]. Cabe mencionar que la escogencia del atributo *nombre* para este experimento fue intencional. En el caso de no haber diseñado *color*

10 El ordenamiento de secuencias de caracteres es mucho más costoso que el ordenamiento de números reales. De allí la explicación de la diferencia de tendencias.

como un atributo difuso, este sería una cadena de caracteres. Por otro lado, aun siendo *color* un atributo difuso, en caso de no usar la opción *START* de la cláusula *ORDER BY*, tendría que ordenarse como cadena de caracteres. Dada la significancia de la interacción entre los factores volumen y tipo de atributo, y el comportamiento antes descrito, se podría concluir que para consultas *ORDER BY*, a la medida que crece el volumen de datos, el desempeño es mejor con atributos difusos que con atributos precisos.

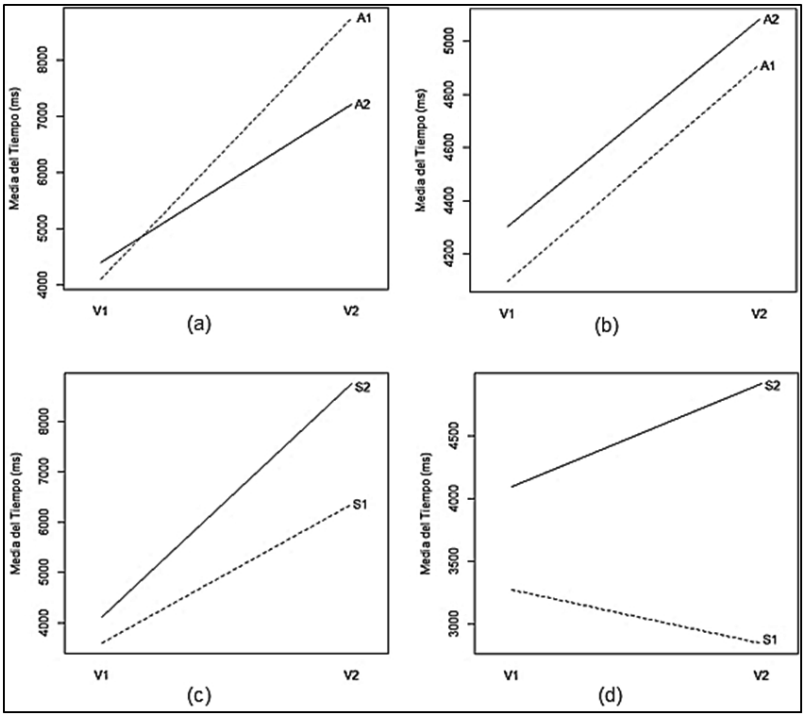

Figura 2. Gráficos de Interacción de factores: (a) Volumen-Atributo en consulta ORDER BY (E1); (b) Volumen-Atributo en consulta GROUP BY (E2); (c) Volumen-SGBD en consulta ORDER BY (E3); (d) Volumen-SGBD en consulta GROUP BY (E4)

Consultas GROUP BY. La ANOVA de la Tabla 4, experimento E_2, muestra que el único factor que tiene cierta significancia en el tiempo de ejecución de las consultas con GROUP BY es el volumen, haciendo pensar que no hay diferencia significativa entre hacer el agrupamiento con un atributo difuso o con un atributo preciso. La Figura 2b indica que el crecimiento del tiempo de ejecución aumenta según el volumen de datos donde la tendencia es similar para ambos tipos de atributos.

Hay una pequeña diferencia entre el tiempo de ejecución de las consultas con atributos difusos que es un poco mayor que el de las consultas con atributos precisos, pero la diferencia parece ser una constante que está un orden de magnitud por debajo de los tiempos observados, lo que puede explicarse por el impacto de la traducción de estas consultas difusas a precisas que involucra el uso de las tablas LABEL y SIMILARITY del catálogo difuso. Se concluyen entonces, que el desempeño de *fuzzy*doDB para consultas con GROUP BY es igual si el agrupamiento es difuso o preciso.

3.2.2 Segundo grupo de experimentos: según SGBD. Con estos experimentos se busca mostrar el impacto de la extensión del SGBD en el tiempo de procesamiento de consultas precisas. La Tabla 3b contiene los resultados de las corridas del segundo grupo de experimentos (Tipo de SGBD).

Consultas ORDER BY. La ANOVA de la Tabla 4, experimento E_3, señala que ambos factores, SGBD y volumen de datos, son muy significativos en el tiempo de ejecución de las consultas con ORDER BY, así como la interacción entre ambos factores. La Figura 2c muestra que el crecimiento del tiempo de ejecución aumenta según el volumen de datos donde la tendencia es más fuerte para el caso del manejador extendido. Hay una diferencia entre el tiempo de ejecución de las consultas sobre el manejador extendido que va aumentando a medida que el volumen crece. Esto se explica por el impacto del procesamiento adicional generado por la extensión. Se concluye entonces, que el SGBD extendido tiene un tiempo de ejecución mayor en el caso de consultas con ORDER BY al de SGBD original.

Consultas GROUP BY. La ANOVA de la Tabla 4, experimento E_4, muestra que el factor más significativo en el caso de consultas con cláusula GROUP BY es el SGBD. La interacción entre el SGBD y el volumen de datos, también tiene significancia, pero en grado menor. La Figura 2d presenta que el crecimiento del tiempo de ejecución en el SGBD extendido aumenta a medida que aumenta el volumen de datos, mientras que en el caso del SGBD original, el tiempo de ejecución disminuye. Esto último pareciera que no tiene sentido, sin embargo, se asume que tal comportamiento se debe a que *PostgreSQL* posee mecanismos de optimización que contrarrestan el efecto del aumento de volumen. Se concluye entonces, que el SGBD extendido tiene un tiempo de ejecución mayor en el caso de consultas con GROUP BY al de SGBD original.

4. Conclusiones

En este trabajo se han reportado los resultados de un primer estudio experimental de desempeño de *fuzzy*doDB, una extensión de *PostgreSQL*, realizada con acoplamiento medio, la cual provee funcionalidades de definición, manipulación y consultas de bases de datos con atributos difusos.

Una de las contribuciones del trabajo realizado fue la creación de una primera base de datos experimental provista de atributos difusos. Esta base de datos es sobre los Pokémon, personajes de la famosa serie de videojuegos. Con esta base de datos se pueden ilustrar las nuevas funcionalidades de *fuzzy*doDB para el tratamiento de atributos difusos. También puede ser una referencia para hacer comparación de SGBD que involucren estos tipos de atributos.

Para el análisis de desempeño se usó un método estadístico formal que busca explicar la influencia de diversos factores considerados. Este método permite hacer un estudio con un número reducido de corridas y pequeña escala de los niveles de los factores. Con los experimentos se obtienen los datos para los cuales se adapta un modelo que describe la variable observada como una combinación lineal de los factores y sus interacciones. Se usó un diseño experimental 2^k que es usual este tipo de estudios. La variable observada fue el tiempo total de ejecución de las consultas.

Se consideraron las consultas con ordenamiento (ORDER BY) y consultas con agrupamiento (GROUP BY). Se realizaron cuatro experimentos organizados en dos grupos. El primero tuvo como propósito estudiar el impacto del uso de atributos difusos *Tipo 3*; mientras el segundo se enfocó en el impacto de la extensión respecto al SGBD original.

Se evidenció que en *fuzzy*doDB, para consultas ORDER BY, a la medida que crece el volumen de datos, el desempeño del ordenamiento difuso basado en atributos *Tipo 3* es mejor que el del ordenamiento preciso para etiquetas. Esto obedece al hecho que se ordena con base en grados de similitud (que son valores numéricos) y no en cadenas de caracteres (con mayor costo de comparación). En el caso de las consultas con GROUP BY, el desempeño de *fuzzydoDB* es igual si el agrupamiento es difuso o preciso. Estos resultados son alentadores para continuar trabajando en dar soporte a atributos difusos.

El segundo grupo de experimentos mostró que la extensión hecha al SGBD degrada su desempeño, debido a la arquitectura de acoplamiento medio que se ha usado para implementar la extensión. En esta arquitectura se colocó un mediador por el cual pasan todas las consultas y sus resultados. Esto está agregando una sobrecarga al procesamiento de la consulta. Se considera de alguna manera que es el precio que se está pagando por tener más funcionalidad. En caso de no querer tener este costo, habría que implementar la extensión con una arquitectura de acoplamiento fuerte, lo cual sería mucho más complejo en términos de desarrollo pues involucra una intervención a código abierto.

El análisis de desempeño de consultas con ordenamiento difuso y consultas con agrupamiento difuso estudio realizado se restringió a atributos *Tipo 3*. Sin embargo, fuzzydoDB también da soporte a atributos *Tipo 2*, *Tipo 4* y *Tipo 5*. Sería interesante realizar nuevos estudios con estos otros tipos de atributos difusos. También por ser un primer estudio, se trabajó con consultas muy sencillas sobre una sola tabla, sin cláusula WHERE y de que solo involucraron un criterio en la cláusula ORDER BY o en la GROUP BY. Se propone para trabajos futuros estudiar el desempeño en caso de consultas más complejas.

Dado que existen bases de datos que están disponibles al público como referencia de comparación para SGBD relacionales, se sugiere que en trabajos futuros se tome alguna de ellas y se analice cómo podría extenderse con atributos difusos, de forma que puedan usarse también en futuros estudios experimentales.

Agradecimientos

Se agradece el apoyo de los estudiantes del curso Miniproyecto de Desarrollo de Software en los últimos tres años, quienes han contribuido con la implementación y pruebas de *fuzzy*doDB. Asimismo a Aquel que previó este trabajo: "*Porque somos hechura suya, creados en Cristo Jesús para buenas obras, las cuales Dios preparó de antemano para que anduviésemos en ellas*" (Ef. 2:10).

Referencias bibliográficas

CARRASQUEL, Soraya; GYOMREY, Andras; MOREAU, Sergio; RODRÍGUEZ, Rosseline; STORNELLI, Bishma, TIMAURY Carlos & TINEO, Leonid (2014). Extensión de MariaDB para ordenamiento y agrupamiento difuso, En: Novática, No. 229 (jul.), Madrid (España): Asociación de Técnicos de Informática. p. 92-97. ISSN: 0211-2124.

CARRASQUEL, Soraya; MONASCAL, Ricardo; RODRIGUEZ, Rosseline & TINEO, Leonid (2016). Processing of Queries with Fuzzy Similarity Domains. In: YAN, Li (ed.) Handbook of Research on Innovative Database Query Processing Techniques. Hershey (USA): IGI Global. p. 88–128. ISBN: 978-1466687677.

CARRASQUEL, Soraya; RODRÍGUEZ, Rosseline & TINEO, Leonid (2013). Consultas con Ordenamiento basado en Similitud. En: Telematique, Vol. 12. No. 1. Maracaibo (Venezuela): Universidad Privada Dr. Rafael Belloso Chacín. p. 24-45. ISSN: 1856-4194.

CARRASQUEL, Soraya; RODRÍGUEZ, Rosseline & TINEO, Leonid (2014). Consultas con Agrupamiento basado en Similitud. En: Ingeniare: Revista chilena de ingeniería. Vol. 22, No. 4 (oct.). Arica (Chile): Universidad de Tarapacá p. 517-527. ISSN: 0718-3305.

CASTEJÓN SANDOVAL, Osiris (2011). Diseño y Análisis de Experimentos con Statistix. Maracaibo (Venezuela): Fondo Editorial Biblioteca Universidad Rafael Urdaneta. 203 p. ISBN: 978-980-7131-14-8.

CORONADO, David; CARRASQUEL, Soraya; MONASCAL, Ricardo; RODRIGUEZ, Rosseline & TINEO, Leonid (2015). Portal de fuzzydoDB. En: Memorias de la Tercera Conferencia Nacional de Computación, Informática y Sistema (26-28/10/2015), Valencia (Venezuela): Sociedad Venezolana de Computación, SVC. p. 328–332. ISBN: 978-980-7683-01-2.

GALINDO, José (2008). FSQL (Fuzzy SQL) A Fuzzy Query Language [en línea]. Málaga (España): Universidad de Málaga. <http://www.lcc.uma.es/~ppgg/FSQL/> [consulta: 20/04/2016]

GALINDO, José; URRUTIA, Angélica & PIATTINI, Mario (2006). Fuzzy Databases: Modeling, Design and Implementation. Hershey (USA): Idea Group Publishing. ISBN 1-59140-324-3

JAIN, Raj (1991). The Art of Computer Systems Performance Analysis. New Jersey (USA): John Wiley & Sons, Inc. 685 p. ISBN: 978-0-471-50336-1.

MEDINA, Juan Miguel; PONS, Olga & VILA, María Amparo (1994). Gefred: A Generalized Model of Fuzzy Relational Databases. In: Information Sciences: Informatics and Computer Science Intelligent Systems Applications, Vol. 76, No. 1-2 (sep), Amsterdam (The Netherlands): Elsevier B.V. p. 87-109. ISSN: 0020-0255.

R CORE TEAM (2015). R: A language and environment for statistical computing [online]. Vienna (Austria): R Foundation for Statistical Computing. <http://www.R-project.org/> [consult: 20/04/2016].

THE POKEMON COMPANY (2011). Pokédex [online]. Bellevue (USA): The Pokemon Company. <http://www.pokemon.com/es/pokedex/> [consult: 20/04/2016].

TIMARÁN, Ricardo (2001). Arquitecturas de Integración del Proceso de Descubrimiento de Conocimiento con Sistemas de Gestión de Bases de Datos: un Estado del Arte, En: Ingeniería y Competitividad, Vol. 3, No. 2. Cali (Colombia): Universidad del Valle. p. 45-55. ISSN 0123-3033

WIKIA, Inc (2014). WikiDex la enciplopedia Pokémon [online]. San Francisco (USA): Wikia. <http://es.pokemon.wikia.com/wiki/WikiDex> [consult: 20/04/2016].

ZADEH, Lofti (1965). Fuzzy Sets. In: Information and Control, Vol. 8, No. 3 (jun). Amsterdam (The Netherlands): Elsevier B.V. p. 338-353. ISSN: 0019-9958.

ZADEH, Lofti (1983). A computational approach to fuzzy quantifiers in natural languages. In: Computer & Mathematics with Applications, Vol. 9, No. 1 (jan). Amsterdam (The Netherlands): Elsevier B.V. p. 149-183. ISSN: 0898-1221.

ZADEH, Lofti A. (1971). Similarity Relations and Fuzzy Orderings. In: Information Sciences: Informatics and Computer Science Intelligent Systems Applications, Vol. 3, No. 2 (apr). Amsterdam (The Netherlands): Elsevier B.V. p. 177-200. ISSN: 0020-0255.

ZADEH, Lofti (2015). Fuzzy logic - a personal perspective. In: Fuzzy Sets and Systems, Vol. 281 (dec). Amsterdam (The Netherlands): Elsevier B.V. p. 4-20. ISSN: 0065-0114.