Revista Técnica de la Facultad de Ingeniería, Universidad del Zulia
ISSN: 0254-0770
revistatecnica@gmail.com
Universidad del Zulia
República Bolivariana de Venezuela

Ana Isabel Aguilera Faraco; Marlene Goncalves Da Silva
Una Implementación para el Agrupamiento Difuso en SQL
Revista Técnica de la Facultad de Ingeniería,
Universidad del Zulia, vol. 44, núm. 1, 2021, pp. 38-56
Universidad del Zulia
Maracaibo, República Bolivariana de Venezuela

ppi 201502ZU4659
Esta publicación científica en formato digital es continuidad de la revista impresa
ISSN 0254-0770 / Depósito legal pp 197802ZU38

Edificio Patrimonial La Ciega, primera sede de LUZ

REVISTA TÉCNICA
DE LA FACULTAD DE INGENIERÍA

Una Revista Internacional Arbitrada
que está indizada en las publicaciones
de referencia y comentarios:

- SCOPUS
- SCIELO
- LATINDEX
- DOAJ
- MIAR
- REDIB
- AEROSPACE DATABASE
- CIVIL ENGINEERING ABTRACTS
- METADEX
- COMMUNICATION ABSTRACTS
- ZENTRALBLATT MATH, ZBMATH
- ACTUALIDAD IBEROAMERICANA
- BIBLAT
- PERIODICA
- REVENCYT

"Post nubila phoebus"
"Después de las nubes, el sol"

LUZ en sus 130 años
de fundación
1891-2021

REVISTA TÉCNICA
DE LA FACULTAD DE INGENIERÍA

UNIVERSIDAD DEL ZULIA

VOLUMEN 44 ENERO - ABRIL 2021 NÚMERO 1

Una Implementación para el Agrupamiento Difuso en SQL

***Ana Isabel Aguilera Faraco*[*1], *Marlene Goncalves Da Silva*[2]**

[1]Escuela de Ingeniería Informática, Facultad de Ingeniería, Universidad de Valparaíso, Valparaíso, C.P. 2340000, Chile
[2]Departamento de Computación y Tecnología de la Información, Universidad Simón Bolívar, Caracas, Venezuela, Apartado 89000, Caracas, Venezuela.
*Autor de correspondencia: ana.aguilera@uv.cl
https://doi.org/10.22209/rt.v44n1a05
Recepción: 13 de abril de 2020 | Aceptación: 20 de octubre de 2020 | Publicación: 01 de enero de 2021

Resumen

Los sistemas de gestión de bases de datos (SGBD) relacionales tienen una gran utilidad en el almacenamiento eficiente de grandes volúmenes de datos. En este sentido, se han propuesto algunas extensiones de los SGBD basadas en la lógica difusa, para mejorar la expresividad de los lenguajes de consulta, entre ellos, el lenguaje SQLf (extensión de SQL que soporta condiciones difusas). Por otra parte, el Group-By es un operador de base de datos ampliamente utilizado en el análisis de datos y en los sistemas de apoyo a la toma de decisiones. En muchos casos, parece útil agrupar los valores según su similitud con un determinado concepto en lugar de establecer la agrupación sobre la base de valores iguales. En este contexto, se ha propuesto una nueva estructura de SQLf denominada Fuzzy Group By (FGB), para apoyar una agrupación basada en particiones difusas. En este trabajo, se incorporó la agrupación difusa en PostgreSQLf, que es una extensión del SGBD PostgreSQL, para el manejo de consultas difusas utilizando el lenguaje SQLf con una arquitectura fuertemente acoplada (directamente en el SGBD). Se pronone un algoritmo basado en un *hash* para evaluar el operador FGB y también se evalúa empíricamente el rendimiento de PostgreSQLf sobre el Benchmark™ TPC-H.

Palabras clave: Fuzzy Group By; PostgreSQLf; arquitectura fuertemente acoplada.

An Implementation for SQL Fuzzy Grouping

Abstract

Relational DataBasemanagement systems (DBMS) have a great utility in the efficient storage of large data volumes. Also, some DBMS extensions based on fuzzy logic have been proposed to improve the expressiveness of query languages. Among which SQLf is an extension of SQL that supports fuzzy conditions. Separately, the Group-By is a database operator widely used in data analysis and decision support systems. In many cases, it seems useful to group values according to their similarity to a certain concept rather than establishing grouping on the basis of equal values. In this context, a new SQLf structure called Fuzzy Group By (FGB) has been proposed to support a grouping based on fuzzy partitions. In this work, we incorporated the fuzzy grouping in PostgreSQLf, which is an extension of the PostgreSQL DBMS for the handling of fuzzy queries using the SQLf language on the basis of a tight coupled architecture, i.e., directly into the DBMS. We have proposed an algorithm based on a *hash* to evaluate the FGB operator and also empirically assessed the performance of PostgreSQLf over the TPC Benchmark™ -H (TPC-H).

Keywords: Fuzzy Group By; PostgreSQLf; tight coupled architecture

Introducción

A pesar del vertiginoso desarrollo de la tecnología de las bases de datos, los sistemas de gestión de bases de datos (SGBD) comunes no permiten la expresión de requisitos graduales de los usuarios porque sufren el problema de la rigidez [1], [2], [3]. En un sistema clásico, los requisitos de usuarios deben expresarse de manera precisa. En este sentido, la rigidez de los sistemas clásicos tiene dos consecuencias principales [4]. No hay discriminación de las respuestas según las preferencias de los usuarios y las respuestas en el límite pueden dejar los resultados fuera. La lógica difusa ofrece nuevos instrumentos para acceder y procesar datos que pueden ser aplicables en sistemas en los que los requisitos de los usuarios no son precisos por naturaleza [5], [6], [7].

El operador estándar de agrupamiento tiene gran importancia para los almacenes de datos y las técnicas de análisis como OLAP y la minería de datos, y tiene propiedades de tiempo de ejecución y escalabilidad relativamente buenas. Aunque la semántica deloperador de agrupamiento es simple, se limita a la igualdad (todas las tuplas de un grupo, tienen exactamente los mismos valores que los atributos de agrupación). Para superar estas limitaciones, Bosc y Pivert [8] proponen ampliar la cláusula Group-By en SQLf mediante la cláusula FuzzyGroup-By (FGB), la cual permite la agrupación basada en particiones difusas predefinidas en los atributos del dominio, en lugar de la igualdad de datos. Una cláusula Group-By en SQL construye una partición basada en los valores (atómicos) de los atributos especificados en esa cláusula, por ejemplo, «GROUP BY A» construye una partición en la que cada grupo está asociado a un valor de A presente en la relación. La idea de Bosc y Pivert [8] es ampliar este mecanismo para construir particiones en términos de intervalos o conjuntos difusos de valores. Además, ellos añaden una variante de la función de agregación de conteo llamada *count-rel*, que calcula la cardinalidad relativa (por ejemplo, el grado de satisfacción medio) asociada a un grupo.

Aplicaciones emergentes, como las bases de datos biológicos y la transmisión de datos, requieren la identificación de grupos de valores aproximados. Adicionalmente, las aplicaciones comerciales con grandes cantidades de datos pueden beneficiarse enormemente de las sentencias SQLf, que identifican grupos de valores similares. La implementación de la agrupación difusa dentro de un motor de base de datos, puede tener la ventaja de que el tiempo de ejecución del operador de Fuzzy Group-By es comparable al del Group-By convencional. Así pues, estetrabajo comprende el desarrollo de una extensión del SGBD PostgreSQL para la gestión de la agrupación difusa, sobre la base de una arquitectura estrechamente acoplada. En una arquitectura estrechamente acoplada [9], [10], [8], todas las tareas, componentes y funcionalidades correspondientes al paradigma de la base de datos a integrar, forman parte del respectivo SGBD como operación primitiva. La principal ventaja de esta arquitectura es que se resuelven todos los problemas de escalabilidad y rendimiento que pueden presentarse en otros tipos de arquitecturas.

Ejemplo de motivación

Considere los datos de la cartelera musical mostrada en la Tabla 1, donde un artículo se caracteriza por un título, año, artista y ventas en millones. Además, considere la consulta de un usuario para determinar la media de ventas por década (sesenta, setenta, ochenta, noventa, etc.). En SQL, esta consulta puede expresarse como[8], [11]: **SELECT label(año), avg(venta) FROM cartelera GROUP BY label(año) USING p(año) = {[1960, 1969], [1970, 1979], [1980, 1989], [1990, 1999], [2000, 2009], [2010,2019]}**; el resultado de esta consulta es presentada en la Tabla 2.

Tabla 1. Cartelera musical.

Título	Año	Artista	Ventas (millones)
Can't Help Falling In Love	1962	Elvis Presley	28
Carnegie Hall Concert	1966	BuckOwens	54
Aretha Franklin: Soul '69	1969	Aretha Franklin	32
SomethingBetterTo Do	1975	Olivia Newton-John	22
Thriller	1983	Michael Jackson	65
ThisIsThe Time	1987	Billy Joel	12
BallerinaGirl	1987	Lionel Richie	53
My Heart Will Go On	1998	CelineDion	8
Hard Candy	2008	Madonna	34
No Line On The Horizon	2009	U2	31
SomeoneLikeYou	2011	Adele	41
LoveYourself	2016	Justin Bieber	23
Cozy Little Christmas	2018	Katy Perry	12

Tabla 2. Valores promedio de ventas de títulos por rango de años.

label (año)	[1960, 1969]	[1970, 1979]	[1980, 1989]	[1990, 1999]	[2000, 2009]	[2010, 2019]
Prom.	38,00	22,00	43,33	8,00	32,50	25,33

Una consulta clásica con una cláusula GROUP BY es la siguiente [3],[9]: SELECT label(A) [, agg, ...] FROM R WHERE ψ GROUP BY label(A) USING p(a)={L_1, ..., L_n}; donde p(A) es una partición definidaen el dominio A, label(A) denota una etiqueta L_i de p(A), y ψ es una condición Booleana o difusa. Si la cláusula WHERE tiene una condición difusa, COUNT solo puede ser utilizado como una función de agregación en la cláusula debido a la dificultad de definir otras funciones de agregación en los conjuntos difusos. Para un L_i dado perteneciente a p(A) y una relación r, se calcula COUNT(L_i). Además, en este trabajo, utilizamos una variante de conteo denominada *count-rel* que calcula la cardinalidad relativa asociada a un grupo, por ejemplo, el grado de satisfacción medio [8], [11]:

$$count(Li) = \sum_{t \in r \wedge t.A \epsilon Li} \mu_{\psi}(t) \quad count - rel(Li) = \frac{\sum_{t \epsilon r \wedge t.A \epsilon Li} \mu_{\psi}(t)}{|\{t \epsilon r \mid t.A \epsilon Li\}|} \quad (1)$$

Para ilustrar el uso de *count-rel*, considerela siguiente consulta: **SELECT label(año), count, count-rel FROM cartelera WHERE ventas=medio GROUP BY label(año) USING p(año) = {[1960, 1969], [1970, 1979], [1980, 1989], [1990, 1999], [2000, 2009], [2010,2019]}**; donde el término medio de venta se define usando el trapecio que se muestra en la Figura 1(b); los cálculos están en la Tabla 3 y los resultados de la consulta están en la Tabla 4; *n* representa $\{t \epsilon r \mid t.A \epsilon Li\}$. Las consultas particionadas pueden extenderse por medio de la lógica difusa. Para ejemplificar las consultas particionadas difusas, considere los trapecios de la Figura 1 y la siguiente consulta que determina el volumen de ventas de todos los títulos más recientes de 1990 para cada clase de ventas (bajo, medio, alto): **SELECT label(ventas), count FROM cartelera WHERE año > 1990GROUP BY label(ventas) USING p(ventas) = {bajo, medio, alto}**.

Sobre la base de los datos de la Tabla 1, el resultado de esta consulta se presenta en la Tabla 5. Es decir, para label (ventas) = alto, la cuenta corresponde a la suma del grado de pertenencia de cada tupla que tiene un año mayor que 1990, son: {MyHeartWillGoOn/0, Hard Candy/0.7, No Line On The Horizon/0.55, SomeoneLikeYou/1, LoveYourself/0.15, Cozy Little Christmas/0}, similarlyforlabel(ventas) = medio and label(ventas) = bajo.

Es importante señalar que en los casos en que la condición es difusa, es necesario redefinir *count y count-rel* de la siguiente manera [8], [11]:

$$count(Li) = \sum_{t \epsilon r} min\left(\mu_{\psi}(t), \mu_{Li}(t.A)\right) \quad count - rel(Li) = \frac{\sum_{t \epsilon r} min\left(\mu_{\psi}(t), \mu_{Li}(t.A)\right)}{\sum_{t \epsilon r} \mu_{Li}(t.A)} \quad (2)$$

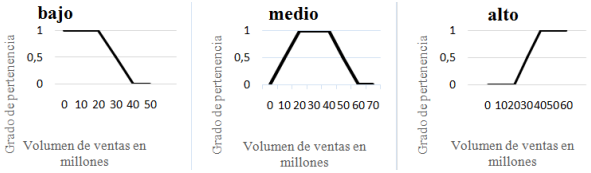

Figura 1. Términos difusos definidos por los trapecios: a) bajo b) medio c) alto.

Tabla 3. Grados de satisfacción calculados para la consulta particionada difusa.

Label (año)	I	Titulo	ventas	$\mu_{\psi}(t)$	$\sum_{t \epsilon n} \mu_{\psi}(t) \frac{\sum_{i \epsilon n} \mu_{\psi}(t)}{\lvert n \rvert}$	
[1960-1969]	1	Can't Help Falling In Love	28	1,00	2,08	0,69
	2	Carnegie Hall Concert	54	0,08		
	3	Aretha Franklin: Soul '69	32	1,00		
[1970-1979]	1	Something Better To Do	22	1,00	1,00	1,00
[1980-1989]	1	Thriller	65	0,00	1,15	0,38
	2	ThisIsThe Time	12	0,60		
	3	BallerinaGirl	53	0,55		
[1990-1999]	1	My Heart Will Go On	8	0,45	0,45	0,45
[2000-2009]	1	Hard Candy	34	1,00	2,00	1,00
	2	No Line On The Horizon	31	1,00		
[2010-2019]	1	SomeoneLikeYou	41	0,95	2,55	0,85
	2	LoveYourself	23	1,00		
	3	Cozy Little Christmas	12	0,6		

Tabla 4. Resultados de la consulta usando una partición definida sobre años.

label (ventas)	[1960, 1969]	[1970, 1979]	[1980, 1989]	[1990, 1999]	[2000, 2009]	[2010, 2019]
count	2,08	1,00	1,15	0,45	2,00	2,55
count-rel	0,69	1,00	0,38	0,45	1,00	0,85

Tabla 5: Resultados de la consulta usando una partición difusa sobre el volumen de ventas.

label (ventas)	Alto	Medio	Bajo
count	2,40	4,95	3,45

Materiales y métodos

PostgreSQLf

En este trabajo, se extendió PostgreSQL basado en una arquitectura de integración estrechamente acoplada, que aprovecha las características internas del SGBD [12], [13], [14]. Desafortunadamente, puede ser necesario un mayor esfuerzo debido a la complejidad del sistema de código abierto y, además, se pierde completamente la portabilidad a la nueva versión. Incluso si la portabilidad se viera comprometida, el interés se centra en el rendimiento y en mostrar la viabilidad de la implementación, como prueba de concepto. Este trabajo es parte de otro gran proyecto llamado PostgreSQLf, iniciado en 2006 [15]. En la Figura 2 se muestran los módulos de PostgreSQL modificados (en color azul). Una consulta ejecutada por PostgreSQL debe pasar por tres módulos: i) el analizador que verifica la validez de la consulta; ii) el planificador que construye el plan de ejecución para la consulta; iii) el ejecutor donde cada consulta es finalmente evaluada.

Catálogo

El catálogo del SGBD es un repositorio en el que se almacenan los metadatos del esquema de la base de datos (información de tablas, columnas, índices, operadores, funciones agregadas, entre otros). PostgreSQL maneja estos metadatos como tablas del sistema. En particular, en este trabajo, el catálogo se amplió para añadir las nuevas funciones de agregación de la agrupación difusa, como parte de las funciones estándar de PostgreSQL.

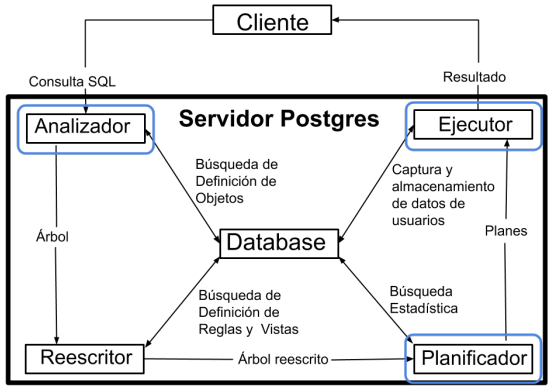

Figura 2.Módulos de PostgreSQL.

Analizador

El analizador comprende dos procesos: el análisis sintáctico y semántico, y el proceso de transformación que toma la instrucción de consulta y la convierte en estructuras de datos operables por PostgreSQL. Los árboles de consulta se gestionan mediante listas enlazadas. El analizador léxico reconoce identificadores, palabras clave SQL, etc. Para las consultas con agrupación difusa, se creó un nuevo tipo de nodo llamado *A_Partition,* con todos los datos relevantes de la partición como su dominio, sus etiquetas, entre otros. Se agregó un puntero (*usingClause*) al nodo *SelectStmt* en una lista de nodos de la partición.

Planificador

La tarea de este módulo es crear un plan de ejecución difuso. Así, combina los posibles caminos de acceso de las relaciones (recorrido de índice, recorrido secuencial o recorrido de índice de mapa de *bits*) y los une si es necesario (unión en bucle anidado, unión en *hash* o unión de ordenamiento por mezcla). La tarea de este módulo es estimar los costos de ejecución de cada método de acceso y elegir el de menor costo. En este trabajo, se verifica el árbol de consulta recibido y luego se crea un plan difuso a partir del árbol de consulta. Se construyen nodos del plan adicionales, para calcular las cardinalidades necesarias para las funciones de agregación de la agrupación difusa.

Ejecutor

Este módulo toma el plan de ejecución del planificador y lo procesa de forma recursiva para obtener el conjunto de filas requerido; utiliza un mecanismo de canalización progresiva por demanda. Cada vez que se llama a un nodo de tipo planificador, este debe enviar una o más filas, o informar de que ha terminado todas las filas requeridas. Para evaluar el operador de Fuzzy Group By, se propone un algoritmo basado en un *hash*. Cada partición representa un recipiente (un conjunto difuso) y la función de *hash* discrimina a qué recipiente pertenece una tupla. El algoritmo 1 muestra el proceso de llenado de la tabla *hash* para la agrupación difusa.

Algoritmo 1. Llenado de la table hash para agrupamiento difuso

```
NuevaTupla ← Ø;
foreachParticionen Consulta do
        foreachEtiqueta en Particiondo
        ifTuplaestá en dominio de Etiqueta then
                NuevaTupla←copiaTupla(Tupla); M ←
GradoSatisfaccionFGB(Tupla, Etiqueta);

EntradaTablaHash ← insertarEnTablaHash(NuevaTupla);
                ifConsulta tiene count_relthen
                N←CardinalidadEtiqueta(Etiqueta);
AvanceAgregados(EntradaTablaHash, M, N);
        elseAvanceAgregados(HashTableEntry, M);
```

En primer lugar, el ejecutor calcula los grados de pertenencia para los predicados que representan las etiquetas de partición en la agrupación difusa. Para las consultas con agrupación difusa o clásica, los grupos se crean sobre la marcha, con base en las etiquetas

especificadas por el usuario. En el archivo *nodeAgg.c* de PostgreSQL se debe modificar la forma en que el sistema agrupa las tuplas, ya que para cada tupla el resultado se agrupa por el valor de uno o más campos normalmente, es decir, si se agrupa por un campo de edad, entonces hay tuplas que se agrupan por edad = 23, otras por edad = 31, etc. En otras palabras, los grupos son conjuntos disjuntos, en los que no hay elementos que se encuentren en más de un conjunto (Figura 3a). Esta clasificación se realiza mediante una tabla de *hash* que es entonces extraída para cada grupo, para aplicar los cálculos finales como funciones de agregación. La diferencia en la agrupación difusa es que cada tupla puede pertenecer a uno o más grupos (Figura 3b); esto ocurre cuando se utilizan etiquetas superpuestas (clásicas o difusas) dentro de la partición. Por lo tanto, para los casos de consultas cuantificadas, la función *agg_fill_hash_table* verifica si cada tupla del proceso pertenece a alguna etiqueta de cada partición especificada en la consulta. Si la tupla pertenece a más de una etiqueta, se duplica y se introduce en la tabla *hash*. Si se trata de una partición difusa o hay predicados difusos en la consulta, se calcula su grado de pertenencia y se utiliza para los cálculos de las funciones de agregación *count_p y count_prel.* En el caso de esta última, será necesario calcular la cardinalidad de cada etiqueta.

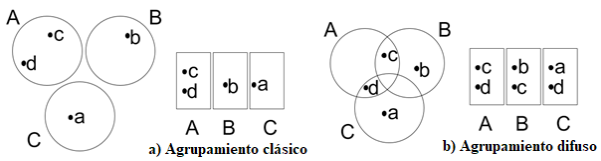

Figura 3. Diferencia entre agrupamientos estándar y propuesta.

La implementación de la función *AvanceAgregados* es exactamente la misma que la función predeterminada de PostgreSQL, excepto que la función *count_paggregate* es una función de suma que añade los grados de membresía como parámetros, y la función *count_preles* igual a una función de suma pero añade la división entre el grado de membresía y la cardinalidad pasadas como parámetros.

Estudio experimental

En esta sección, se estudió el rendimiento de las consultas de agrupamiento difuso dentro de PostgreSQL. Primero, se describe la prueba de rendimiento, la métrica y los detalles de implementación del estudio experimental.

Prueba de rendimiento

TPC-H proporciona un esquema de base de datos, un generador de datos y consultas para evaluar el rendimiento de un sistema en condiciones estándar. Los tamaños de los conjuntos de datos eran de 1 y 5 GB. En este estudio experimental, se consideraron las tablas *part* (200.000/1.000.000 filas), *partsupp* (800.000/4.000.000 filas) y *supplier* (10.000/50.000 filas). El generador de datos se aplicó solo para consultas con agrupación difusa. Así, se definieron 24 consultas: i) seis consultas con condición Booleanay partición clásica; ii) seis consultas con condición difusa y partición clásica; iii) seis consultas con condición Booleana y partición difusa; iv) seis consultas con condición difusa y partición difusa. También, se definieron seis consultas clases equivalentes.

Métricas de evaluación

El rendimiento fue medido y reportado como el tiempo total de ejecución (el tiempo transcurrido en milisegundos entre el envío de una consulta a PostgreSQL y la entrega de las respuestas). El tiempo se midió usando el SQL EXPLAIN ANALYZE.

Implementación

Se realizaron experimentos usando PostgreSQL 8.2 sobre Ubuntu Desktop 10.10, arquitecturaAMD64, equipadocon un Intel Core 2 Duo T6400 y 4 GB RAM.

Resultados y discusión

A continuación se realiza un análisis exploratorio de los datos, mediante estadísticas descriptivas. Para ello, se trazó un histograma de tiempo (ms) en la Figura 4. Según el conjunto de datos, la frecuencia más alta se encuentra en el segundo intervalo (entre ≈333 y ≈666 ms) y la distribución de los datos se asemeja a una función logarítmica normal (Figura 4 izquierda). Como medida de la ubicación, se tomo, en este caso fue de 1.034,76739 ms. También hay una variabilidad muy alta de los datos, ya que la desviación estándar fue de 959,035129 ms. En cuanto a la medida de la dispersión, la diferencia entre los valores mínimo y máximo uee alta (101,477 y 3.900,827 ms).

La Figura 4 (centro), contiene un diagrama de caja para el tiempo total de ejecución. Considerando el tipo de consulta de la Figura 4 (derecha), se puede observar que las consultas Booleanas clásicas exhibieron los tiempos más altos. Aunque las consultas clásicas devolvieron las mismas tuplas y se agruparon de la misma forma que la extensión, utilizan el operador UNION que genera una carga de trabajo adicional que aumenta el tiempo de ejecució. Así pues, la solución demuestra experimentalmente, que fue más eficaz para las consultas de agrupamiento. Además, hubo elemento atípico que está relacionado con el alto volumen de datos devueltos.

Posteriormente, se realizó un ANOVA para determinar si existía una diferencia significativa entre las medias de la muestra (Tabla 6). Al respecto, se puede decir, con un nivel de significancia del 95% (α= 0,05), que hubo una diferencia al variar el tamaño de la base de datos o el tipo de consultas. Es decir, el volumen de la base de

datos y el tipo de consultas, afectan significativamente al tiempo de ejecución de la consulta.

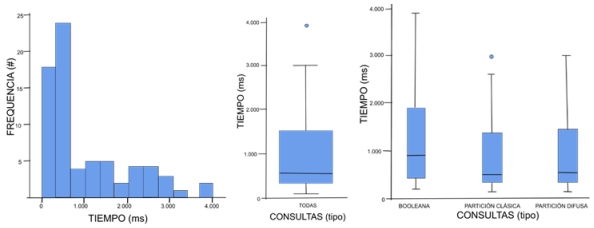

Figura 4. Histograma del tiempo total de ejecución (izquierda). Diagrama de caja para el tiempo de ejecución total (centro). Diagrama de caja por tipo de consulta (derecha).

Tabla 6. Resultados ANOVA.

Origen	Suma de cuadrados de tipo III	*gl*	Media cuadrática	*F*	Sig.
Modelo corregido	40.534.397.077	11	3.684.945.189	8.927	0,000
Volumen	31.168.599.521	1	31.168.599.521	75.506	0,000
Volumen*tipo de consulta	861.253.945	2	430.626.972	1.043	0,359
Número* tipo de consulta	2.321.883.863	2	1.160.941.931	2.812	0,068
Error	24.767.737.862	60	412.795.631		
Total	142.395.670.474	72			
Total corregido	65.302.134.939	71			

Trabajos relacionados

Los grupos difusos son uno de los campos de las matemáticas que usan la teoría de conjuntos difusos, presentada por Rosenfeld [16]. También se ha utilizado en varios ámbitos de aplicación, como: clasificación, reconocimiento de patrones, procesamiento de imágenes, inteligencia artificial, sistemas de información, análisis de datos, toma de decisiones y agrupación de bases de datos. Zhang y Huang [17] propusieron algunas instrucciones SQL para permitir la agrupación en el contexto de los datos espaciales. Básicamente, estas instrucciones actúan como envoltorios de los algoritmos de agrupación convencionales, pero no se estudia una mayor integración con las bases de datos. Li [18] extendió el operador GROUP BY para agrupar todas las tuplas aproximadamente dentro de un número de grupos predefinidos. Este marco utiliza algoritmos de agrupación convencionales, por ejemplo, K-means, y aplica índices de mapa de *bits* para integrar la agrupación y la clasificación en las bases de datos. Silva *et al*. [19] propusieron un Group-By basado en un principio de similitud en PostgreSQL. La verde [20] introdujo un nuevo operador capaz de producir grupos de mayor calidad para varios dominios de datos. En este trabajo, la atención fue centrada en el agrupamiento difuso basado en conceptos vagos, en lugar de un agrupamiento basado en la similitud. Tampoco se basó en el descubrimiento de grupos porque los mismos se especifican explícitamente en la consulta mediante particiones difusas. Además, en los trabajos citados anteriormente no consideraron una extensión de las consultas difusas como SQLf, sino solo una extensión de una funcionalidad SQL particular.

Haciendo referencia en la similitud basada en grupos por construcciones, Bosc y Pivert [8] propusieron cómo introducir la agrupación de datos en términos de conceptos vagos (predicados difusos) dentro de las sentencias SQLf [8], [11]. Por último, se han desarrollado algunas otras implementaciones de agrupación difusa. Un programa de *windows* llamado Fuzzy Grouping ofrece tres métodos de agrupación difusa a los ecologistas de la comunidad [21]. La transformación de agrupación difusa es una técnica utilizada para realizar tareas de limpieza de datos, mientras se eliminan los datos duplicados [22]y es parte de *Microsoft* SQL *Server* [23].

Conclusiones

En este trabajo, el SGBD PostgreSQL se amplió para ejecutar consultas de agrupamiento difuso dentro de una arquitectura estrechamente acoplada. Hasta donde se sabe, esta es la primera implementación de este tipo. Para las consultas con agrupación difusa, el catálogo se amplió para añadir las nuevas funciones de agregación a la agrupación difusa como parte de las funciones estándar de PostgreSQL. Las nuevas funciones añadidas fueron *count_p* y *count_prel*. La extensión propuesta involucró la modificación de varios módulos del gestor de base de datos, estos fueron el analizador, el planificador y el ejecutor. Cabe destacar que este método se puede aplicar a otras extensiones que se quieran realizar más allá del paradigma difuso. Los cambios involucran incorporación de nuevas estructuras y la implementación de nuevos operadores que gestionen estas nuevas estructuras y las operaciones asociadas a ellas. En particular, para evaluar el operador Fuzzy Group By, propusimos un nuevo algoritmo basado en un hash. El algoritmo implementó el proceso de llenado de una tabla hash para el agrupamiento difuso. Así mismo, a medida que se añadían nuevas palabras clave a la sintaxis y se añadían nuevos nodos para el árbol Parser, se introdujo una nueva forma de agrupar las tuplas mediante etiquetas de partición (clásica o difusa).

Este trabajo se centró en la agrupación difusa basada en conceptos vagos, en lugar de la agrupación

enfocada en la similitud. No se focalizó en el descubrimiento de grupos ya que los mismos se especifican explícitamente en la consulta, mediante particiones difusas. El estudio experimental muestra que el mecanismo de agrupación propuesto, integrado en un motor de base de datos, es más eficiente que otras soluciones. La carga reside en el uso de las funciones de agregación implementadas, específicamente el caso de la función *count_prel*, en la que es necesario ejecutar consultas adicionales para los cálculos de cardinalidad. Por lo tanto, cuanto mayor sea el volumen de registros que tenga la tabla, más carga se añade al tiempo de ejecución.

Como trabajo futuro, se pueden mencionar dos puntos clave a considerar y permitir funciones de agregación difusa más completas: la implementación de la agrupación difusa definida por Bosc y Pivert [8], y el soporte de la cláusula HAVING para la evaluación de la consulta difusa, además de implementar la función de agregación *count-g*.

Agradecimientos

Se agradece al profesor Ralph Grove, un amigo en Norfolk, VA quien ayudó con la edición en inglés de este trabajo.

Referencias bibliográficas

[1] Bosc P. y Pivert O.:"SQLf: a relational database language for fuzzy querying". IEEE Transactions on Fuzzy Systems, 3 (1), (1995)1-17. https://doi.org/10.1109/91.366566.

[2] Bosc P. y Pivert O.:"SQLf Query Functionality on Top of a Regular Relational Database Management System". Studies in Fuzziness and Soft Computing, (2000), 171-190.

[3] George R., Petry F. E., Buckles B. P. y Srikanth R.:"Fuzzy database systems—challenges and opportunities of a new era". Int J of Intelligent Systems, Vol. 11, No. 9, (1996), 649-659.

[4] Pivert O.:"Contribution à l'interrogation flexible de bases de données: expression et évaluation de requêtes floues". Doctoral dissertation, Université de Rennes 1. (1991).

[5] Galindo J., Urrutia A. y Piattini M.:"Representation of Fuzzy Knowledge in Relational Databases". Fuzzy-Databases: Modeling, Design and Implementation, (2006), 145-170.

[6] Goncalves M. y Tineo L.:"SQLf3: an extension of SQLf with SQL3 features". In Proceedings of 10th IEEE International Conference on Fuzzy Systems, (2001), 477-480.

[7] Sanchez, H.R., Sarango, D.E. y Cucuri,M.I.: "Evaluación de un sistema de alimentación avícola basado en lógica difusa".Revista Técnica de Ingeniería Universidad del Zulia, Vol. Especial, No. 1, (2020), 3-10.

[8] Bosc P. y Pivert O.:"On a fuzzy group-by clause in SQLf". International Conference on Fuzzy Systems, (2010), 1-6.

[9] Aguilera A., Cadenas J.T. y Tineo L.:"Fuzzy Querying Capability at Core of a RDBMS".InAdvances in Data Mining and Database Management, IGI Global.Hershey, 2011, 160-184.

[10] Bosc P. y Galibourg M.:"Indexing principles for a fuzzy database". J. InformationSystems, Vol. 14, No. 6, (1989), 493-499.

[11] Pivert O. y Bosc P.:"Fuzzy Group By". In: Fuzzy preference queries to relational databases. WorldScientific, (2012), 251–265.

[12] 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", Vol. 3, No. 2, (2001), 45-55.

[13] Smits G., Pivert O. y Girault T.: "ReqFlex: fuzzy queries for everyone". Proc. VLDB Endow., Vol. 6, No. 12, (2013), 1206-1209.

[14] Aguilera A., Cadenas J. y Tineo L.:"Rendimiento de Consultas SQLf en arquitecturas débil y fuertemente acopladas". Revista Multiciencias, Latindex Venezuela, Vol. 11, No 4, (2011), 410-415.

[15] Cadenas. J.:"Una contribución a la interrogación flexible de bases de datos: Optimización y evaluación a nivel físico", Master Thesis, USB, Caracas, Venezuela. (2006)

[16] Rosenfeld A.:"Fuzzy groups". Journal of mathematical analysis and applications, Vol. 35, No. 3, (1971), 512-517.

[17] Zhang C. y Huang Y.:"Cluster By: a new sql extension for spatial data aggregation". In Proceedings of the 15th annual ACM International Symposium on Advances in Geographic Information Systems,(2007), 1-4.

[18] Li C., Wang M., Lim L., Wang H. y Chang K. C.:"Supporting ranking and clustering as generalized order-by and group-by". In Proceedings of the ACM SIGMOD International Conference on Management of data, (2007), 127-138.

[19] Silva Y. N., Aref W. G. y Ali M. H.:"Similarity group-by". In Proceeding of 2009 IEEE 25th International Conference on Data Engineering, (2009), 904-915.

[20] Laverde N. A., Cazzolato M. T., Traina A. J. y Traina C.:"Semantic Similarity Group By Operators for Metric Data". In Similarity Search and Applications (SISAP), Vol. 10609, (2017), 247-261.

[21] Henderson P.A., Seaby R.M.H. y Somes J.R.:"Fuzzy Grouping". Pisces Conservation Ltd., Lymington, Hampshire, UK.,Vol. 2, (2014).

[22] Pudło F. y Ząbkowski T.:"Information Quality improvement methods in Management Information Systems". InformationSystems in Management II, Wyd. SGGW, (2008), 124-133.

[23] Zhang J., Guyer C., Milener G. y Petersen T.:"Fuzzy Grouping Transformation". https://docs.microsoft.com/en-us/sql/integration-services/data-flow/transformations/fuzzy-grouping-transformation?view=sql-server-2017, (2017).

UNIVERSIDAD
DEL ZULIA

REVISTA TECNICA

DE LA FACULTAD DE INGENIERIA
UNIVERSIDAD DEL ZULIA

Vol. 44. N°1, Enero - Abril 2021, pp. 04 - 58

Esta revista fue editada en formato digital y publicada en Diciembre de 2020, por el ***Fondo Editorial Serbiluz, Universidad del Zulia. Maracaibo-Venezuela***

www.luz.edu.ve
www.serbi.luz.edu.ve
www.produccioncientificaluz.org