Procesamiento de Consultas Vagas sobre Datos Difusos: Principio de Derivación

Claudia González, Leonid Tineo

Universidad Simón Bolívar, Departamento de Computación,
Apartado 89000, Caracas 1080-A, Venezuela
{claudia,leonid}@ldc.usb.ve

Resumen. Los DBMS fallan en la representación de datos imperfectos y el procesamiento de consultas vagas que involucran preferencias de usuarios. Lenguajes como FSQL y SQLf usan los Conjuntos Difusos para subsanar tales problemas. Sin embargo el problema no es sólo semántico, sino también práctico, pues se deben proveer mecanismos eficientes de procesamiento para tales características. SQLf que sólo permite consultas vagas sobre datos precisos. En trabajos previos se ha concebido el Principio de Derivación que ha demostrado mantener bajo el costo añadido del procesamiento de SQLf. Sin embargo, para consultas vagas sobre datos difusos, como son las de FSQL, no se han concebido mecanismos eficientes de procesamiento. En este trabajo nosotros extendemos el Principio de Derivación para el procesamiento de consultas vagas sobre datos difusos, como parte de una propuesta de integración de FSQL y SQLf en un solo estándar para Bases de Datos Difusas.

Palabras Claves: Bases de Datos Difusas, SQLf, FSQL, Procesamiento de Consultas, Principio de Derivación.

1 Introducción

Los sistemas informáticos se enfrentan cada vez más con data incompleta, redundante, dispersa y diluida que no puede ser ignorada porque se considera muy útil para satisfacer los requerimientos de los usuarios los cuales además tienen tendencia a ser vagos involucrando preferencias. Estos problemas provienen de la misma naturaleza del hombre, que es dicotómico, ambiguo y que al expresarse por naturaleza también es ambiguo y poco concreto. Por otro lado los Sistemas Manejadores de Bases de Datos tradicionales no están capacitados para interpretar este tipo de información y requerimientos. Como respuesta a este problema, existen algunas propuestas para Bases de Datos Difusas como: — OMRON [1], — FQUERY [2], — ISKREOT [3], — FSQL [10] [11] [4], — SQLf [13][14] y otros [12][5][6]. Entre éstas, SQLf es el lenguaje más rico en la expresión de consultas vagas (difusas) y FSQL en la representación de data imperfecta (difusa). Ambos lenguajes son extensiones de SQL con la aplicación de la Teoría de Conjuntos Difusos.

XXXIV Conferencia Latinoamericana de Informática (CLEI 2008), pp. 739-748, 2008.

La integración de FSQL y SQLf en un solo estándar para Bases de Datos Difusas sería de gran valor en la expansión del impacto de los esfuerzos hecho en esta área, así como su inclusión en los DBMS existentes y su uso en el desarrollo de aplicaciones. En un trabajo previo [16] se estudian y comparan las componentes difusas incluidas en ambos lenguajes y en [15] se presenta un estándar de consultas difusas se presenta un instrumento que selecciona criterios ampliamente conocidos para el diseño de lenguajes, mencionados en [7][8], a saber: Poder Expresivo, Facilidad de Uso para Novatos, Escritura Intuitiva, Ortogonalidad, Minimalidad, Facilidad de Implementación, Eficiencia en la Evaluación. Como resultado del trabajo previo, se propuso un estándar integrado que incluye, entre otros aspectos la representación y manipulación de datos y relaciones difusas.

En la actualidad SQLf no incluye el procesamiento de datos difusos. Para SQLf se ha concebido previamente el Principio de Derivación [9], el cual permite evaluar una consulta difusa cobre el resultado de una consulta precisa derivada, de manera que se mantiene lo más bajo posible el costo añadido por el procesamiento de consultas difusas. Por otro lado, FSQL sí contempla la representación y procesamiento de atributos difusos, más no relaciones difusas. Lamentablemente, hasta la fecha, el procesamiento de FSQL se hace con una estrategia ingenua pues no se ha concebido mecanismos eficientes.

Apuntando hacia la Facilidad de Implementación y Eficiencia en la Evaluación que son criterios deseables en el diseño de lenguajes, en este trabajo nosotros proponemos la extensión del Principio de Derivación para el procesamiento de consultas vagas sobre datos difusos. Aunque hay una amplia gama de posibles consultas en esta clase, nos concentramos en el caso más representativo que es cuando: — El dato difuso es representado como un rango impreciso, es decir un conjunto difuso con función de membresía trapezoidal; y — La consulta vaga usa sólo un predicado lingüístico reprensado también mediante una función de satisfacción trapezoidal. Este caso resulta ser la condición difusa atómica más básica, sin embargo de suficiente complejidad para ser considerada como alcance de este trabajo.

El resto del presente trabajo artículo se organiza como sigue: — En Sección 2 se presentan los Aspectos Semánticos; — La Sección 3 define el principio del Excelente Compilador; — En la Sección 4 se determinan las condiciones necesarias y suficientes para determinar que dos Trapecios Posiblemente Iguales; — En la Sección 5 se termina de aplicar el Principio de Derivación Insertando un α-corte; — Finalmente, la Sección 6 puntualiza las Conclusiones y Trabajos Futuros.

2 Aspectos Semánticos

En su definición original SQLf no permite el almacenamiento de atributos difusos mientras que FSQL sí. Para la integración de estos lenguajes en un único estándar [15], cuatro tipos de atributos difusos se han propuesto. Éstos son ligeramente diferentes de los de FSQL. Los tipos propuestos son: — *Tipo 1:* atributos con datos precisos sobre un dominio ordenado. Los predicados difusos son definidos sobre estos. Ejemplo: la edad tiene un dominio ordenado entre [0,1000] años y sobre ésta se definen etiquetas como: joven, anciano, etc. — *Tipo 2:* atributos con datos precisos,

pero con un dominio no ordenado. Las relaciones de similitud y los predicados difusos pueden ser definidos sobre estos. Ejemplo: los colores no son ordenados, pero el verde es más similar al azul que al rojo. — *Tipo 3:* datos difusos con un dominio ordenado. Se puede definir una edad como joven cuando está entre 0 y 30 años de edad y adulta para 31 a 60 años, por lo que claramente joven<adulto. — *Tipo 4:* datos difusos con un dominio no ordenado. También se puede definir un nuevo color como {0.5/naranja, 0.5/rosado} y una relación de similitud. La sintáxis para especificar el tipo de datos de un atributo en la instrucción CREATE/ALTER TABLE será: `NOMBRE TIPODIFUSOn TIPOCLASICO`, donde `NOMBRE` es el nombre del atributo, `n` es el número del tipo de atributo difuso y `TIPOCLASICO` es el dominio básico.

El razonamiento Posibilístico provee dos medidas de verdad para las proposiciones lógicas: posibilidad (Π) y necesidad (N). En condiciones difusas sobre datos clásicos (crisp), caso de SQLf, la medida de necesidad resulta inútil [16], porque esta provee los mismos resultados que la medida de posibilidad. Por ello, SQLf usa solo la medida de posibilidad. Por otro lado, FSQL envuelve datos y condiciones difusas. Entonces, FSQL permite le uso de ambas medidas posibilidad o necesidad según la preferencia del usuario. Sin embargo, debido a que el enfoque flexible favorece las consultas que envuelven relaciones difusas definidas por preferencias del usuario. En la integración de FSQL y SQLf [15] se propone el uso de la medida de posibilidad como medida por defecto y la posibilidad de permitirle al usuario la especificación de la selección de la medida de necesidad.

A pesar que existen diversas estructuras de extensiones difusas a SQL, este nos enfocaremos en la estructura básica de un bloque de consulta:

```
SELECT <expresiones_de_atributos> FROM <relaciones>
WHERE <condiciones_difusas> THOLD <α>
```

La respuesta de esta consulta es un multiconjunto difuso de registros en el producto cartesiano de las relaciones en la cláusula del FROM que satisfaga la condición difusa con la medida de posibilidad mayor o igual que el nivel de tolerancia *(THOLD)* α, el grado de membresía de el resultado para cada registro está determinado por la medida de posibilidad.

Se restringe el alcance de nuestro estudio en este artículo a las condiciones difusas de la forma $T_1=T_2$, siendo T_1 un atributo difuso de tipo 3 cuyo valor es un conjunto difuso con forma de trapecio cuyo núcleo(T_1)=[b,c] y suporte(T_1)=[a, d] y T_2 un predicado definido por un conjunto difuso con función de membresía con forma de trapecio teniendo núcleo(T_2)=[f,g] y suporte(T_2)=[e,h]. Intuitivamente la semántica de la condición $T_1=T_2$ es: "El dato T_1 cumple el predicado T_2" o, lo que es equivalente, "Los trapecios T_1 y T_2 son posiblemente iguales". El grado de verdad de $\mu(T_1=T_2)$ se define en la ecuación (1).

$$\mu(T_1=T_2)=\sup_{x\in U,\, y\in U} \min\left(\Pi_{T_1}(x), \Pi_{T_2}(y), \mu(x,y)\right) \qquad (1)$$

3 Excelente Compilador

Las implementaciones de los lenguajes difusos se han hecho sobre manejadores de bases de datos existentes. Por lo que, hasta ahora, en la mayoría de las implementaciones se ha utilizado un acoplamiento débil [17], que consiste en la construcción de una capa lógica que traduce las consultas difusas en precisas [5]. Una vez evaluada la consulta precisa, se computa el grado de satisfacción del registro en la respuesta. Para un lenguaje de bases de datos, el tener un excelente compilador permitirá al motor de del manejador el producir mejores planes de evaluación. En este sentido, para SQLf se ha concebido el Principio de Derivación, el cual establece que dada una consulta difusa puede derivarse una consulta precisa que obtiene exactamente las mismas filas (derivación fuerte) o, en el peor de los casos un conjunto resultado con un número limitado de filas indeseadas (derivación débil). La aplicación de este principio consiste en procesar la consulta difusa sobre el resultado de la consulta precisa derivada, de manera de mantener bajo el número de accesos a filas y cálculo de grado de satisfacción de condiciones difusas. Para obtener la consulta derivada se utiliza el concepto de α-corte de un conjunto difuso, el cual consiste en el conjunto preciso de todos los elementos del universo que pertenecen al conjunto difuso con un grado de membresía mayor o igual al nivel α. El Principio de Derivación consiste en expresar el α-corte de una consulta difusa en una consulta que incluya únicamente términos y expresiones precisas.

El principio de derivación define una condición a ser concatenada en la cláusula del WHERE de la consulta precisa que es traducida la consulta difusa. Esta condición evita el cómputo del grado de los registros que no aparecerán en la respuesta. La aplicación de este principio a las consultas en SQLf ha sido el objeto de trabajos previos [9]. En [18] se presenta el uso del α-corte para el procesamiento de consultas difusas envueltas en los operadores de álgebra relacional de SQLf2. En [19] se presenta la extensión para cuantificadores predefinidos, consultas recursivas, vistas, subconsultas y el constructor CASE. Por lo que se espera que la extensión de este principio para datos difusos representados como trapecios produzca grandes beneficios en el desempeño de las consultas difusas. Con el propósito de aplicar el Principio de Derivación al tipo de consultas de nuestro interés en este trabajo, es necesario derivar una condición booleana $DNC(T_1 = T_2, \leq, \alpha)$ que sea equivalente a $\mu(T_1 = T_2) \geq \alpha$ para generar la consulta derivada (derivación fuerte):

```
SELECT <attribute expressions> FROM <relations>
WHERE DNC(T1=T2,≤, α)
```

4 Trapecios Posiblemente Iguales

El problema de verificar que un predicado representado como un trapecio es posiblemente igual a un atributo difuso representado también como trapecio o que dos atributos difusos representados como trapecios son posiblemente iguales, se reduce a

conseguir el punto de altura máxima en la intersección de 2 trapecios T1(a,b,c,d) y T2(e,f,g,h), recuérdese que la medida de posibilidad es $\sup_{x \in X} \min(\Pi_{T_1}(x), \Pi_{T_2}(x))$.

Hasta ahora en SQLf y FSQL los trapecios utilizados tienen altura 1. Mantener esta suposición simplifica nuestro problema porque lo reduce a hallar las intersecciones generadas translación del trapecio T2 sobre el dominio del trapecio T1. Esta translación genera las siguientes posibilidades: a) los núcleos coinciden de T1 y T2 en al menos un punto, b) T2 corta a T1 por la derecha, c) T2 corta a T1 por la izquierda.

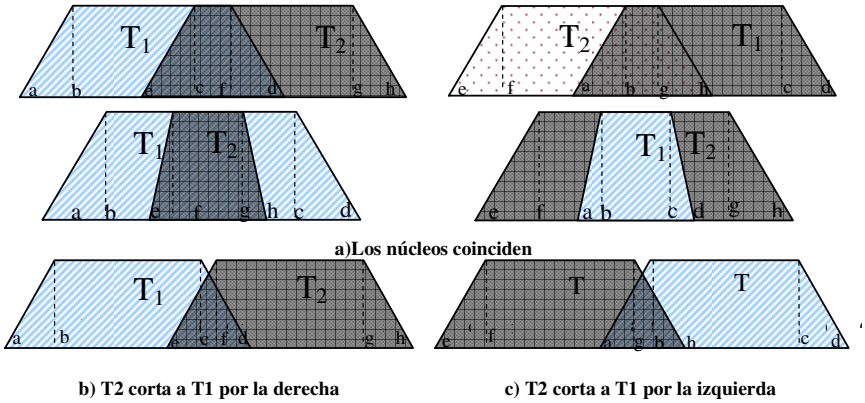

Fig. 1: Posibles intersecciones de dos trapecios de la misma altura.

La condición necesaria y suficiente para afirmar que los núcleos coinciden en al menos un punto sería: $(b \leq f \wedge f \leq c) \vee (b \leq g \wedge g \leq c) \vee (f \leq b \wedge b \leq g)$.

A continuación se presentan las formas en que nos interesa el corte que le hace T2 a T1 por la derecha. En la gráfica sólo se presenta el lado que nos interesa de T2, su lado izquierdo.

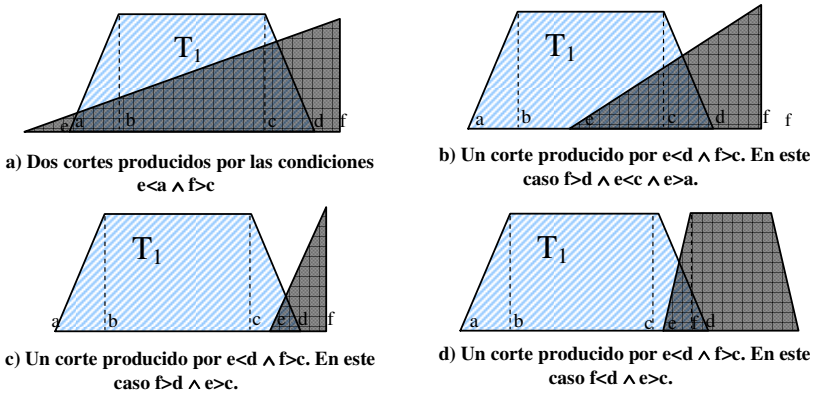

Fig. 2: Posibles intersecciones de dos trapecios de la misma altura.

Existen diversas condiciones pero las que se satisfacen para todos los casos son e<d ∧ f>c. Ésta es una condición necesaria y suficiente para caracterizar el hecho de que los trapecios se cruzan de este modo. Se puede afirmar que la condición e<d es necesaria para que ocurra un corte derecho, cuando e>d ya no ocurre como lo muestra la siguiente figura:

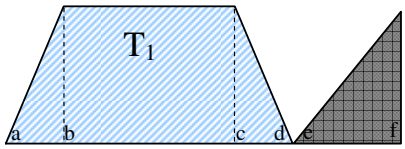

Fig. 3: Trapecios sin intersección e>d

La combinación de las condiciones e<d ∧ f>c ∧ e≤f tendrá como consecuencia que las rectas formadas por los puntos (e,0), (f,1) tengan la siguiente figura.

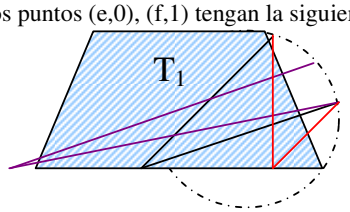

Fig. 4: Intersección de los segmentos de rectas $\overline{(e,0)(f,1)}$ y $\overline{(c,1)(d,0)}$

La condición e<d ∧ f>c garantiza que los segmentos de recta $\overline{(e,0)(f,1)}$ y $\overline{(c,1)(d,0)}$ se cruzan en un punto. En este caso la medida de posibilidad es la coordenada y del punto de corte.

Vale destacar que podría existir un corte por el lado izquierdo como se muestra en la Fig. 2 a, sin embargo el corte más interesante es el derecho, porque es el corte con mayor altura. Adicionalmente, la condición c<f también implica que los trapecios no se tocan en sus núcleos.

Finalmente, el caso en que T2 corta a T1 por el lado izquierdo es análogo a éste, en ese caso a<h y b>g.

5 Insertando un α-corte

Cuando se inserta un α-corte se añade la condición de que los trapecios se intersecten en algún punto cuya coordenada en el eje de las y sea mayor que el α-cut. El máximo α-corte es 1, en ese caso estaríamos solicitando la coincidencia de los núcleos de los trapecios. Para cualquier otro caso bastaría con verificar que el punto perteneciente a la intersección con mayor altura sea mayor que el α-corte.

De forma similar a como se hizo anteriormente dividamos el problema en a) los núcleos coinciden de T1 y T2 en al menos un punto, b) T2 corta a T1 por la derecha, c) T2 corta a T1 por la izquierda.

Para el caso a), la intersección ocurre cuando la coordenada de las y es 1, por lo tanto el punto de intersección siempre será mayor o igual que el α-corte.

En caso que T2 corte a T1 por la derecha, para b) el punto de intersección de mayor altura será precisamente donde se cruzan las rectas formadas por los puntos *(e,0), (f,1)* de T_2 y *(c,1),(d,0)* de T_1.

Dados los puntos {*(c,1), (d,0)}* pertenecientes a la *recta 1: y=mx+p*, la pendiente sería $m = \frac{(1-0)}{(c-d)} = \frac{1}{c-d}$. p de la ecuación se calcula evaluando la recta *y=mx+p* en el punto *(d,0).*

$$\Rightarrow \quad \left(y = \frac{x}{(c-d)} + p\right)$$

<*Evaluando en (d,0)*>

$$\left(0 = \frac{d}{(c-d)} + p\right) \equiv \left(p = \frac{d}{(d-c)}\right)$$

De forma análoga a partir de los puntos *{(e,0),(f,1)}* y la recta 2: *y=nx+q* se obtiene $n = \frac{1}{f-e}$; $q = \frac{e}{(e-f)}$

La intersección ocurrirá en un punto común entre las rectas y para hallarlo basta igualar las ecuaciones de las rectas. Sin embargo la coordenada que nos interesa encontrar es el de las y, porque lo que buscamos es que y≥α. Para hallar la coordenada de las y, igualamos las ecuaciones en el eje =de las x. Despejando x en ambas ecuaciones de la recta se obtiene $x = \frac{y-p}{m}; x = \frac{y-q}{n}$

$$\left(\frac{y-p}{m} = \frac{y-q}{n}\right) \equiv (ny - np = my - mq) \equiv ((n-m)y = np - mq)$$

$$\equiv \quad y = \frac{np - mq}{n-m}$$

≡ <*Sustituyendo p, q, n, m*>

$$y = \frac{\frac{d}{(f-e)(d-c)} - \frac{e}{(e-f)(c-d)}}{\frac{1}{f-e} - \frac{1}{c-d}}$$

$$\equiv \quad y = \frac{\frac{d}{(f-e)(d-c)} - \frac{e}{(f-e)(d-c)}}{\frac{(c-d)-(f-e)}{(f-e)(c-d)}}$$

$$\equiv \quad y = \frac{\frac{d-e}{(f-e)(d-c)}}{\frac{(c-d)-(f-e)}{(f-e)(c-d)}} = \frac{\frac{d-e}{(f-e)(d-c)}}{\frac{(f-e)-(c-d)}{(f-e)(d-c)}}$$

$$\equiv \quad y = \frac{d-e}{(f-e)-(c-d)}$$

El punto de corte en la coordenada *y* queda definido como la distancia entre los puntos d y e: *d(d,e)*, entre la diferencia de las distancias *d(f,e)*, *d(c,d)*.

Análogamente se obtiene que el punto de corte cuando T2 corta a T1 por la izquierda es: $y = \frac{(a-h)}{(g-h)-(b-a)}$

Ahora se puede concluir que verificar que un predicado representado como un trapecio es posiblemente igual a un atributo difuso representado también como trapecio o que dos atributos difusos representados como trapecios son posiblemente iguales, considerando un α-corte, se reduce a verificar la condición:

$$(b \le f \wedge f \le c) \vee (b \le g \wedge g \le c) \vee (f \le b \wedge b \le g) \vee \qquad (2)$$

$$DNC(T_1 = T_2, \ge, \alpha) \equiv \left(e < d \wedge c < f \wedge \frac{d-e}{(f+d)-(e+c)} \ge \alpha\right) \vee \left(a < h \wedge g < b \wedge \frac{a-h}{(g+a)-(h+b)} \ge \alpha\right)$$

Finalmente, se deine el grado de satisfacción $\mu(T_1 = T_2)$ como se muestra en (3). Claramente el cómputo de esta función es de orden constante O(1) y nos permite en la evaluación de la consulta reemplazar la expresión (1) cuyo orden es $O(|U|^2)$.

$$\mu(T_1 = T_2) = \begin{cases} 1 & Si\ b \le f \le c \vee f \le g \le c \\ \frac{(d-e)}{(f-e)-(c-d)} & Si\ e < d \wedge c < f \\ \frac{(a-h)}{(g-h)-(b-a)} & Si\ a < h \wedge g < b \\ 0 & En\ otro\ caso \end{cases} \qquad (3)$$

De esta manera queda demostrado en teoría la aplicabilidad del Principio de Derivación para el procesamiento de consultas vagas sobre datos difusos para el caso en que tanto el predicado como el dato sean representados mediante conjuntos difusos definidos por funciones de membresía trapezoidal.

6 Conclusiones y Trabajos Futuros

En este trabajo se presenta el criterio de excelente compilador usado en el diseño de lenguajes. Este criterio nos que un excelente compilador nos guiará a producir mejores planes de evaluación, para lo cual resulta útil la extensión del principio de la derivación para el procesamiento de datos difusos, que evitarán el cómputo del grado de membresía de los registros que no pertenezcan a la respuesta. Como una primera aproximación se define su extensión para los datos difusos con representación más compleja: los datos representados como trapecios, se provee la definición de la condición a ser incluida en la consulta traducida de SQL difuso a SQL estándar y la definición del grado de satisfacción de los registros. Esto se considera gran aporte en la evaluación de consultas flexibles, debido a que evita el cálculo del grado de satisfacción a los registros que no aparecerán en la respuesta y mientras mayor sea la selectividad de los predicados mayor será el beneficio debido a que se calculará el grado de satisfacción a menos registros. Adicionalmente, también se presenta una función de membresía cuya evaluación es de orden constante. El tener un excelente compilador que permita un procesamiento eficiente de consultas vagas sobre datos difusos sin duda será una clave para la popularización de las bases de datos difusas en el desarrollo de los sistemas de información que requieren este tipo de bondades. En un trabajo futuro se proveerá un estudio experimental que permita mostrar en la práctica los beneficios en cuanto a desempeño de las consultas que utilizan este principio con respecto a las que no lo usan y además se proveerá la extensión del mismo para datos definidos por extensión. También se tiene previsto el proponer en el futuro un mecanismo de evaluación que integre el Principio de Derivación al núcleo de un DBMS mediante una arquitectura de acoplamiento fuerte que tiene ventajas en cuanto a escalabilidad y desempeño.

Agradecimientos. Este trabajo es soportado en parte por la Fundación Venezolana Gubernamental para la Ciencia, Innovación y Tecnología, Proyecto G-2005000278. Nosotros, Claudia y Leonid, alabamos al Señor Jesucristo por su asombrosa gracia y las fuerzas que el nos dio durante la realización de este trabajo.

7 Referencias

1. H. Nakajima, T. Sogoh and M. Arao. “Fuzzy Database Language and Library-Fuzzy Extension to SQL”, Proceedings of Second Fuzz-IEEE, pp 477-482, 1983.
2. J. Kacpryzyk and S. Zadrozny. “Fuzzy Queries in Microsoft AccessTM v.2”, Proceedings of Fuzzy IEEE’95 Workshop on Fuzzy Database Systems and Inf. Retrieval, pp 61-66, 1995.
3. G. Loo and K. Lee. “An Interface to Databases for Flexible Query Answering: A Fuzzy-Set Approach”. LNCS 1873.
4. 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.
5. Z.M. Ma and Li Yan, “Generalization of Strategies for Fuzzy Query Translation in Classical Relational Databases. Information and Software Technology, Volume 49, Issue 2, 2007
6. M. Wong and K. Leung. “A fuzzy Database-Query Language”. Information Systems. Vol 15. No. 5, pp 583-590, 1990.

7. Ghezzi, C and Jazaheri, M. Programming Language Concepts. John Wiley & Sons, Inc. New York, N. Y. USA, 1997.
8. Scott, M: Programming Language Pragmatics, Morgan Kaufmann Publishers, 2005.
9. Bosc, P., Pivert, O., "On the efficiency of the alpha-cut distribution method to evaluate simple fuzzy relational queries", Advances in Fuzzy Systems-Applications and Theory, Vol. 4, Fuzzy Logic and Soft Computing, B. Bouchon-Meunier, R.R. Yager, L.A. Zadeh Eds, Wold Scientific, Pp. 251-260, 1995.
10. Medina, J., Pons, O. & Vila M., GEFRED: A Generalized Model of Fuzzy Relational Databases, Information Sciences, Vol. 77, N° 6, (1994), Pp. 87-109..
11. Galindo J., Urrutia A., Piattini M., "Fuzzy Databases: Modeling, Design and Implementation". Idea Group Publishing Hershey, USA, 2006.
12. M. Umano, S. Fukami, M. Mizumoto, K. Tanaka, "Retrieval Processing from Fuzzy Databases", Technical Reports of IECE of Japan, Vol 80, No 204 (on Automata and Languages), pp. 45-54, AL80-50 (1980).
13. Goncalves, M., Tineo, L.: "SQLf Flexible Querying Language Extension by means of the norm SQL2", Proceedings of the 10th IEEE International Conference on Fuzzy Systems Fuzz-IEEE 2001, Vol. 1, Melbourne, Australia,
14. Goncalves, M., Tineo, L.: "SQLf3: An Extension of SQLf with SQL3 Features", Proceedings of the 10th IEEE International Conference on Fuzzy Systems Fuzz-IEEE 2001, Vol. 1, Melbourne, Australia, (2001
15. Galindo J, González C, Tineo L. "Fuzzy Database Integration using Expressive Power". FSKD'08. Fuzzy Systems and Knowledge Discovery, October 2008. (To appear).
16. González, C., Urrutia, A., Tineo, L. "FSQL and SQLf: Towards a Standard in Fuzzy Databases". In J. Galindo (Ed.), Handbook of Research on Fuzzy Information Processing in Databases. Information Science Reference, Idea Group Inc. ISBN: 9781599048536.
17. Timarán, R. Arquitectura del SGBD PostgreSQL. Una visión general. Universidad de Nariño, Departamento de Sistemas, San Juan de Pasto, Colombia. Documento presentado en el Seminario de Base de Datos USB, Noviembre 2006.
18. Goncalves, M., Tineo, L.. "Derivation Principle in SQLf2 Algebra Operators". Proceedings of the First International Conference on Fuzzy Information Processing Theories and Applications. Volumen: 1. Volumen: 1. Key: 7-302-062994. Páginas: 453 - 459. 2003.
19. Goncalves, M., Tineo, L., **"**Derivation Principle in Advanced Fuzzy Queries**"**. The 14th IEEE International Conference on Volume , Issue, 25-25. (2005) Page(s):579 – 584 Digital Object Identifier 10.1109/FUZZY.2005.1452458