Incorporando consultas difusas en el desarrollo de software

Adding fuzzy queries to software development

Marlene Goncalves, PhD., Rosseline Rodríguez, MSc. y Leonid Tineo, PhD.
Universidad Simón Bolívar, Caracas - Venezuela
{mgoncalves,crodrig,leonid}@ldc.usb.ve

Recibido para revisión 14 de Abril de 2009, aceptado 23 de Octubre de 2009, versión final 27 de Noviembre de 2009

Resumen—**Los sistemas de información tradicionales se basan en consultas precisas a bases de datos. Sin embargo muchos requerimientos de usuarios en la realidad involucran términos vagos del lenguaje natural cuya semántica varía según el contexto y las preferencias del usuario. En este sentido, se han extendido los lenguajes de consultas a bases de datos incorporando la lógica difusa. No obstante, estos avances aún no han sido incorporados en las metodologías de desarrollo usadas en los sistemas de información. En este artículo, se propone una extensión a las metodologías para el desarrollo de software que permite incorporar los beneficios de la lógica difusa para expresar requerimientos que involucran preferencias de usuarios. Adicionalmente, en este trabajo, se muestra la aplicación de los nuevos aspectos metodológicos a través del desarrollo de un caso de estudio real sobre la encuesta de opinión estudiantil de la Universidad Simón Bolívar.**

Palabras Clave—**Consultas Difusas, Desarrollo de Software, Manejo de Preferencias, Metodología, Requerimientos Vagos, SQLf.**

Abstract—**In this paper we present an alternative of gap measurement by means of an optical sensor, when it is used in the magnetic levitation process. Non linear relation between current and voltage in the coil, and the gap value, is used for doing gap measure indirectly. Design and training results of selected neural net, which learns the pattern among variables, are presented. In addition, the scope of a new self-sensor application is explained. It depends on the time required for updating.**

Keywords—**Fuzzy Queries, Methodology, Software Development, SQLf, User Preferences Handling, Vague Requirements.**

I. INTRODUCCIÓN

Con el deseo de dar una representación a las preferencias de usuarios en requerimientos que involucran elementos lingüísticos de naturaleza vaga, han surgido algunas extensiones el estándar SQL añadiéndole lógica difusa, tales como SQLf[3], FSQL[5] y SoftSQL[2], entre otros. SQLf es el más completo de ellos por la variedad de consultas que permite el uso de condiciones difusas y por haber evolucionado con los estándares de SQL hasta la versión de SQL2003[8]. A lo largo de los últimos siete años, haciendo uso de SQLf, se han tenido diversas experiencias en el desarrollo de aplicaciones con consultas difusas[7]. Éstas se han realizado utilizando modelos de desarrollo de software usuales conocidas, que no tienen previsión alguna para la incorporación de consultas difusas.

Un modelo de desarrollo de software es la representación formal o simplificada del proceso de desarrollo de software. El proceso de desarrollo de software requiere por un lado un conjunto de conceptos, una metodología y un lenguaje propio. A este proceso también se le llama el ciclo de vida del software. El ciclo de vida de software se utiliza para estructurar las actividades que se llevan a cabo en el desarrollo de un producto de software[12].

Existen varios de estos modelos de desarrollo que, a pesar de presentar marcadas diferencias, han sido aceptados mundialmente y han permanecido como estándares de desarrollo, debido a su practicidad y a su notable garantía que respalda el éxito del desarrollo de sistemas. Entre estos modelos se destacan: el modelo de desarrollo en cascada [17], el modelo en espiral [4], la programación extrema [1], el proceso de desarrollo iterativo [12], entre otros.

El objetivo de este trabajo es el extender los modelos de desarrollo de software para que se adapten a la creación aplicaciones que requieran consultas difusas sobre bases de datos.

Este documento se ha estructurado en secciones como sigue: ¾ La sección II está dedicada al contexto de este trabajo sobre Modelos de Desarrollo de Software. ¾ La sección III presenta los Fundamentos de SQLf, el lenguaje de consultas difusas que se usa para la implementación de aplicaciones. ¾ La sección IV contiene la contribución principal de este trabajo, la Propuesta Metodológica para la incorporación de consultas difusas en los modelos de desarrollo de software. ¾ La sección V muestra cómo utilizar los elementos metodológicos añadidos, a través de un Caso de Estudio, la realización de una aplicación para consultar la Encuesta de Opinión Estudiantil de la Universidad Simón Bolívar. ¾ La sección VI puntualiza las Conclusiones del trabajo realizado.

II. MODELOS DE DESARROLLO DE SOFTWARE

El modelo en cascada es un modelo de un proceso de desarrollo y su filosofía es completar cada etapa con un alto grado de exactitud, antes de iniciar la siguiente. El punto crítico es que una etapa no ha terminado hasta que la documentación y/u otros productos asociados con esa etapa hayan sido completados. Este modelo fue introducido por Winston Royce en la década de 1970 [17].

El modelo en cascada presenta varias desventajas: 1) los proyectos reales raramente siguen el flujo secuencial que propone el modelo; 2) puede ser difícil para el cliente establecer explícitamente al principio todos los requisitos e inclusive pueden cambiar los requisitos iniciales; 3) hasta llegar a las etapas finales del proyecto, no estará disponible una versión operativa del programa, por lo que un error importante no detectado hasta que el programa esté funcionando puede ser desastroso; 4) algunos integrantes del equipo de desarrollo pueden tener que esperar que otros integrantes completen las tareas pendientes de una etapa anterior. Ante estas desventajas surge el modelo V, o cascada modificado que permite la retroalimentación y el solapamiento entre fases. A diferencia del modelo en cascada, el modelo V es un modelo iterativo y no lineal. El modelo V fue creado en Alemania, en el año 1992 por el Ministerio de Defensa [17].

Las etapas que siguen el modelo en cascada y el modelo V son la factibilidad, el análisis de requerimientos, diseño, codificación, integración, prueba, implantación y mantenimiento. En la etapa de factibilidad se define un concepto referente al producto de software y se determina su factibilidad dentro del ciclo de vida del software. En la etapa de análisis de requerimientos se analizan las necesidades de los usuarios finales del software para determinar qué objetivos debe cubrir el sistema. En la etapa de diseño se descompone y organiza el sistema en elementos que puedan elaborarse por separado, aprovechando las ventajas del desarrollo en equipo. En la etapa de codificación se construye un conjunto completo y verificado de componentes de programas. En la etapa de integración, se ensamblan los componentes de programas como un sistema completo y se comprueba que el sistema funciona correctamente antes de ser puesto en producción. La etapa de prueba se centra en los procesos lógicos internos del software y en la detección de errores, asegurando que todas las sentencias se han comprobado. En la etapa de implantación, se instala el sistema y se entrena a los usuarios del mismo. Finalmente, en la etapa de mantenimiento se corrigen errores o se introducen mejoras al sistema.

Por otra parte, el modelo de desarrollo en espiral nace con el objetivo de captar lo mejor del modelo V y los prototipos, incorporando el análisis de riesgos. Este modelo forma parte de los modelos conocidos como evolutivos y fue propuesto por Boehm en 1986 según [4].

Las actividades de este modelo son una espiral, donde cada bucle es una actividad. Sin embargo las actividades no están fijadas a priori, sino que son elegidas dependiendo del análisis de riesgo. La primera vuelta alrededor de la espiral podría resultar en el desarrollo de una especificación del producto; sucesivas iteraciones podrían ser usadas para desarrollar un prototipo y progresivamente versiones más sofisticadas del software. De esta manera en cada vuelta se van desarrollando nuevas versiones del software, cada vez más completas, donde las especificaciones son paulatinamente resueltas y así se continúa hasta llegar al sistema final. Aunque las etapas pueden ser las mismas en cada vuelta a la espiral, las actividades son mayoritariamente definidas en el análisis de riesgo de la etapa anterior.

Por otro lado, la reutilización de sistemas de aplicaciones se ha utilizado por muchos años en las compañías de software que implementan subsistemas en varias máquinas y los ajustan a diversos entornos. La ingeniería de software basada en reutilización es un enfoque de desarrollo que trata de maximizar la reutilización de software existente. La reutilización de funciones se ha llevado a cabo por medio de bibliotecas estándar de funciones reutilizables como las bibliotecas de gráficos y matemáticas. Sin embargo, aunque ha existido interés en la reutilización de componentes desde principios de los 80, hasta hace pocos años ha sido aceptado como un enfoque práctico para el desarrollo de sistemas de software [12].

A diferencia del modelo en cascada, los modelos iterativos se basan en dividir el proyecto de desarrollo en varias etapas, llamadas iteraciones. Las iteraciones son cortas y su duración es fija. La idea central es que, en cada una de esas iteraciones, se construye una parte pequeña del sistema; esto se llama a veces «desarrollo incremental». Para esa parte del sistema, se realiza todo el proceso: análisis, diseño, programación y pruebas. La iteración termina con un ejecutable que incluye todas las partes del sistema construidas hasta el momento. Los aspectos del sistema con más riesgo se construyen en las primeras iteraciones. A pesar que muchos ven el modelo de desarrollo iterativo como una práctica moderna que surge en reemplazo del modelo de cascada, su aplicación data desde mediados de 1950 [12].

Existen varios modelos de desarrollo iterativos. Entre ellos se destaca el Proceso Unificado y su variante el Proceso Unificado de «Rational», que son estándares actuales en el ámbito internacional [9].

El proceso unificado comprende cuatro grandes fases: concepción, elaboración, construcción y transición. La concepción define el alcance del proyecto y desarrolla un caso de negocio. La elaboración define un plan del proyecto, especifica las características y fundamenta la arquitectura. La construcción crea el producto y la transición transfiere el producto a los usuarios.

Un modelo iterativo nuevo que ha surgido con fuerza últimamente es *eXtreme Programming* (XP) o Programación Extrema. La programación extrema es una metodología de desarrollo de software ágil propuesta por Kent Beck, autor del primer libro sobre la materia [1]. Así como otros procesos ágiles, también conocidos como metodologías livianas, intenta evitar los tortuosos y burocráticos caminos de las metodologías tradicionales enfocándose en la gente y los resultados.

En la programación extrema, el cliente está en el pleno derecho de hacer cuantos cambios necesite al proyecto si con ello consigue un mejor resultado final. Para lograr esto, desarrolladores y clientes deben trabajar en conjunto y muy de cerca desde el primer día y las metas en términos de características, tiempos y costos deben ser reajustas permanentemente.

La metodología se basa en la idea de ciclos, diferenciando dos tipos de ciclos: El ciclo de Negocio y el ciclo de Desarrollo. Los ciclos de Negocio giran alrededor de actividades como comunicados de prensa, entrenamiento, cobro y entrega de software; dichos ciclos se denominan *release* (liberación). Los ciclos de Desarrollo son mucho más cortos que los ciclos de Negocio, dentro de ellos se realizan las operaciones de análisis de requerimientos, diseño, toma de decisiones y desarrollo y se denominan *iteration* (iteración). Para ayudar al entendimiento de este sistema por parte de los clientes, se posee otro tipo de medición -formado por uno o varios *iterations*-, el cual es el *story* (historia). Cada *story* representa una funcionalidad que el cliente desea en el producto final.

Finalmente, todos estos modelos de desarrollo de software se caracterizan por algún tipo de análisis, diseño e implementación. El análisis es una actividad requerida en cualquier modelo de desarrollo de software donde se identifica el problema, se documentan los requerimientos, se involucran a los usuarios y expertos del dominio de aplicación y pueden crearse prototipos. El diseño constituye un refinamiento del análisis; en el diseño se enriquece la descripción del análisis incorporando aspectos de la plataforma de desarrollo, se define la arquitectura del sistema y se diseñan los componentes del sistema. En la implementación, los componentes son desarrollados en algún lenguaje de programación seleccionado.

Sin embargo, ninguno de estos modelos de desarrollo considera consultas difusas. Así, en este trabajo, se extienden las actividades principales de los modelos de desarrollo de software (análisis, diseño e implementación) con el fin de soportar consultas difusas durante el desarrollo del software.

III. FUNDAMENTOS DE SQLF

Los conjuntos difusos fueron introducidos Zadeh en 1965 [20], su intención era modelar clases imprecisas en Sistemas de Control. Con el tiempo, los conjuntos difusos han sido usados en una gran variedad de aplicaciones, fundamentalmente en el Área de Inteligencia Artificial, sin embargo su aplicabilidad en Bases de Datos y Sistemas de Información no ha sido aún completamente explotada[6].

En un conjunto difuso, cada elemento está provisto de un grado que representa su membresía al conjunto. Estos grados inducen un orden que define preferencias. La membresía se define como una función cuyo rango es el intervalo real [0,1]. La función de membresía de un conjunto difuso F es denotada con el símbolo m_F. El soporte de un conjunto difuso es el conjunto clásico de los elementos con $m_F(x)^1 0$. El núcleo es el conjunto que contiene todos los elementos con $m_F(x)=1$. El borde es el conjunto de los elementos que no están completamente excluidos pero tampoco están completamente incluidos $m_F(x)^1 0$ y $m_F(x)^1 1$.

La teoría de Conjuntos Difusos es la base de la Lógica Difusa. En esta lógica, el valor de verdad de una condición (o grado de satisfacción) está en el intervalo real [0,1]. El valor 0 se entiende como «completamente falso», y el valor 1 es «completamente cierto». El valor de verdad de una proposición «s» se denota como m(s). En esta lógica se introduce el término de variable lingüística que es un dato cuyo valor puede ser susceptible de una representación mediante un conjunto difuso, tales como «edad», «estatura», «calificación», «cantidad», «temperatura» y otras medidas por el estilo.

Esta lógica permite dar una interpretación a elementos lingüísticos vagos, también conocidos como términos difusos, a saber: ¾ Predicados, que son los componentes atómicos de esta lógica, definidos por una función de membresía (o conjunto difuso). Éstos corresponden a la clase de términos que se conocen en la literatura como «etiquetas lingüísticas», sin embargo en este trabajo, como en toda todos los trabajos previos sobre SQLf, aquí se denominarán «predicados difusos». ¾ Modificadores, tales como los adverbios, la negación y el antónimo, términos que permiten definir predicados modificados por medio de transformaciones sobre funciones de membresía. ¾ Comparadores, una clase de predicados difusos definido sobre pares de elementos, ellos establecen una comparación difusa. Suelen definirse sobre una medida de distancia como la resta o el cociente o por extensión listando para cada par su grado de membresía ¾ Conectores, operadores definidos para la combinación de condiciones difusas. La negación, conjunción y disyunción difusas son extensiones de sus equivalentes clásicas. ¾ Cuantificadores, términos que describen cantidades, tales como «la mayoría», «cerca de la mitad», «alrededor de 20», éstos son una extensión de los cuantificadores usuales universal y existencial.

SQLf es una extensión de SQL que permite el uso condiciones de búsqueda en lógica difusa. Estas condiciones involucran variables lingüísticas y términos difusos que expresan preferencias de usuario. SQLf está dotado de las construcciones adecuadas para la especificación de las distintas clases de términos difusos antes descritos[13][11].

Por ejemplo: En la Universidad Simón Bolívar de Caracas Venezuela, las calificaciones de los estudiantes en sus cursos son números enteros en la escala del 1 al 5. En este contexto, un usuario podría interesarse en caracterizar las calificaciones mediante el predicado difuso «regular». Este sobre las calificaciones. Este predicado podría interpretarse mediante un conjunto difuso definido por extensión cuya función de membresía es como la graficada en la Figura. 1. La definición por extensión sólo puede hacerse en caso de universos finitos. Por cada elemento que esté en el soporte del conjunto difuso, se da explícitamente el grado de membresía. Esta definición se expresa en SQLf mediante la instrucción:

CREATE FUZZY PREDICATE *regular* **ON** 1..5

AS { 0.5/2, 1.0/3, 0.5/4 }

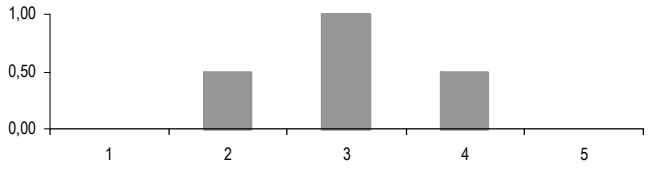

Figura 1. Función de Membresía para el predicado «regular» sobre calificación.

Cabe destacar que la definición del término expresa las preferencias del usuario y puede variar según el contexto. En SQL las órdenes que empiezan con la palabra clave CREATE pertenecen al grupo de comandos de manipulación del esquema. Los objetos que se crean, como es el caso de los predicados difusos, pertenecen a un esquema de usuario particular. La forma de identificar el esquema en el que se está creando el objeto es mediante el nombre del esquema que se coloca compro prefijo al nombre del objeto, separándolos con un punto. Cuando en la orden no se especifica el esquema., por defecto se toma el esquema de usuario con el cual se abrió la sesión actual de ejecución de SQL. En el ejemplo precedente no se especifica el esquema, por lo que el predicado difuso «regular» se está creando dentro del esquema del usuario que inició la sesión, es un objeto que le pertenece a este usuario pues se trata de su propia preferencia. En SQLf, la orden de creación de un predicado difuso contiene la palabra clave ON que indica el dominio sobre el cual se está definiendo el predicado. Esta es la noción de contexto que maneja SQLf. En el ejemplo el dominio es el rango entero 1..5.

Si el usuario estuviera describiendo calificaciones de otra institución probablemente usaría un rango diferente, tal vez el 1..20 que es muy usual. Si el contexto o dominio es un intervalo de números decimales (continuo), ya no es posible usar una definición por trapecio. Por ejemplo si en lugar de describir las calificaciones individuales, el predicado difuso «regular» se definiera sobre los promedios de calificaciones en la misma escala, el dominio (según el contexto) sería el intervalo [1,5]. En ese caso tendría que usarse una función tipo trapezoidal como la que se muestra en la Figuraura. 2. Este tipo de funciones de membresía es de uso frecuente en sistemas difusos debido a su sencillez. Para especificarla basta con dar los valores de los extremos de los intervalos que definen el soporte y el núcleo del conjunto difuso. Esta definición se expresa en SQLf mediante la instrucción:

CREATE FUZZY PREDICATE *regular* **ON** [1,5]

AS (2.50, 3.25, 3,75, 4,25)

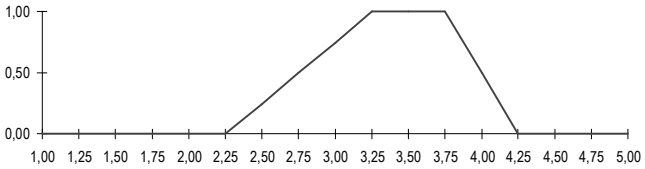

Figura 2. Función de Membresía para el predicado «regular» sobre promedio

Es importante resaltar que aquí se han dado dos definiciones diferentes del predicado difuso «regular». Esto se hecho con el fin de mostrar diferentes formas de definir predicados difusos de acuerdo al contexto y preferencias de usuarios. Sin embargo,

dado que un predicado difuso es un objeto perteneciente a un esquema, no se permite que en un mismo esquema hayan dos definiciones diferentes. Dos esquemas de usuarios diferentes sí podrían tener definiciones diferentes.

La estructura básica de consulta de SQLf es el bloque multirelacional, cuya forma es:

SELECT <atts> **FROM** <rels> **WHERE** <FuzzyCond> **WITH CALIBRATION** [k|a|k,a]

El resultado de esta consulta es el conjunto difuso de filas con los atributos proyectados de la cláusula SELECT en el producto cartesiano de las relaciones en la cláusula FROM que satisfacen la condición difusa de la cláusula WHERE. Como es un conjunto difuso, cada elemento está dotado de un grado de satisfacción y los elementos cuyo grado es cero no forman parte de la respuesta. La cláusula WITH CALIBRATION es opcional, ésta indica la escogencia de las mejores respuestas. Se han propuesto dos tipos de calibraciones: ¾ Calibración Cuantitativa, que indica la escogencia de las mejores k respuestas, de acuerdo a su grado de satisfacción. ¾ Calibración Cualitativa que indica la escogencia de las respuestas cuyo grado de pertenencia es mayor o igual a un nivel mínimo de satisfacción especificado a.

Una variante de esta estructura es el bloque particionado, cuya forma más sencilla es:

SELECT <expresiones> **FROM** <relaciones>

WHERE <CondiciónBooleana>

GROUP BY <atributos>

HAVING <CondiciónDifusa>

WITH CALIBRATION [k|a|k,a]

Una característica interesante de SQLf es que permite el uso de cuantificadores lingüísticos en la condición difusa de la cláusula HAVING. El cuantificador lingüístico es un tipo de función de agregación que se aplica sobre el resultado de la evaluación de una condición difusa a las filas que forman parte de un grupo determinado por la cláusula GROUP BY. De acuerdo a la definición original de SQLf [3], la aplicación del cuantificador en SQLf se hace mediante la palabra clave ARE.

Por ejemplo, el requerimiento de usuario «cuáles son los departamentos que en la mayoría de las notas de los estudiantes tienen regular» se puede expresar en SQLf como:

SELECT departamento **FROM** notas

GROUP BY departamento

HAVING *laMayoria* **ARE** calificacion = *regular*

WITH CALIBRATION 0.25

Aquí definición del cuantificador «laMayoría» podría ser la de la Figura. 3, cuya expresión en SQLf sería:

CREATE FUZZY QUANTIFIER *laMayoría*

AS (0.3333, 0.8333, 1.0, **INFINITE**)

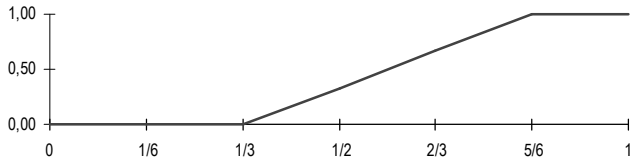

Figura 3. Función de Membresía para el cuantificador «laMayoria»

La semántica de la cuantificación difusa no es trivial. Yager [18] propuso una interpretación adecuada pero no completa. Luego Tineo [14] propuso una semántica completamente ajustada al problema de consultas en bases de datos. El procesamiento de la cuantificación difusa en SQLf es algo complejo y ha sido objeto de un trabajo previo [15].

Existen otras estructuras de consulta en SQLf que no se explican aquí, el lector interesado es referido a la bibliografía [3][7][8].

IV. PROPUESTA METODOLÓGICA

SQLf ha sido usado en el desarrollo de diversas aplicaciones[7][11]. Con base en estas experiencias, aquí se propone una extensión de las metodologías de desarrollo de software de manera que faciliten al desarrollo de aplicaciones con requerimientos vagos que involucran preferencias de usuarios. Aplicaciones que serían desarrolladas utilizando SQLf como lenguaje de consultas a bases de datos. Las extensiones se describen según las fases del modelo del ciclo de vida del software, modelo éste que está implícito en todas las metodologías existentes para el desarrollo de Sistemas de Información. Las extensiones son necesarias sólo en Análisis, Diseño e Implementación.

A. Análisis

En el análisis, se especifican los requerimientos resaltando aquellos conceptos que podrían definir variables lingüísticas y los distintos términos difusos asociados a ellas, como predicados, comparadores, cuantificadores, modificadores y conectores. La presencia de estos conceptos indica que es posible aplicar lógica difusa en la representación de tales requerimientos.

Dentro de esta misma fase de análisis hay que evaluar cada una de las características que permiten detectar si la aplicación necesita del uso de lógica difusa en la expresión de algunos de sus requerimientos de información, o, dicho de otra manera, si la aplicación debe realizar consultas difusas a bases de datos. Estas características se han tomado de un trabajo previo, a efectos de este documento

se utilizan los términos y conceptos propuestos en la bibliografía consultada [11], a saber: ¾ Intuitividad *corresponde a la cualidad del sistema de ser intuitivo para los usuarios del mismo. En las consultas se observan elementos importantes de naturaleza vaga que son muy cercanos al lenguaje natural más que valores, rangos o formulaciones complejas propias de lenguajes de consultas clásicas. Ellas deben ser fácilmente entendibles y manejables por el usuario final. ¾* Flexibilidad *se define como la capacidad del sistema o aplicación de incluir en las respuestas valores en los bordes que pueden ser de interés para el usuario de acuerdo a su preferencia aunque no cumplan de una manera rígida con el ideal buscado. Es decir, el resultado de las consultas difusas puede incluir respuestas que serían rechazadas por una consulta precisa. ¾* Vaguedad *engloba la imprecisión o incertidumbre que en general está presente en los sistemas de conocimiento y de razonamiento humano. Se incluye cuando en los sistemas las consultas abarcan proposiciones que pueden ser parcialmente ciertas o parcialmente falsas. La información manejada puede estar caracterizada por la incertidumbre, lo cual incluye datos imprecisos o consultas vagas que reflejan preferencias de los usuarios. ¾* Tolerabilidad *que es la capacidad de establecer niveles de tolerancia dentro de la imprecisión, estableciendo así rangos aceptables del grado de satisfacción de las respuestas obtenidas por el sistema. Esto es conocido en SQLf como la calibración. ¾* Adaptabilidad*, es decir que el manejo de las preferencias sea parametrizable o adaptable por el usuario. Es decir, cada término difuso debe ser susceptible a cambios debido a percepciones de los diferentes usuarios o contextos. ¾* Gradualidad *es importante cuando las respuestas a los requerimientos de las aplicaciones son discriminadas por grados, de acuerdo con la satisfacción de las condiciones involucradas. Cada elemento o registro en el conjunto de respuestas resultantes de una consulta tiene asociado un valor que expresa su importancia o preferencia para el usuario de acuerdo al requerimiento expresado. ¾* Gerencialidad *se define como la cualidad de un sistema de dar soporte a la toma de decisiones. Aunque las consultas difusas pueden ser útiles para personas en cualquier nivel de la organización o para usuarios comunes, ellas toman mayor aplicabilidad e importancia en el entorno gerencial, donde los términos difusos aparecen con mayor frecuencia y tienen mayor necesidad de ser definidos. El principio de incompatibilidad de [19] lo describe así «en la medida que crece la complejidad de un sistema en esa misma medida decrece la capacidad de escribir enunciados precisos sobre su funcionalidad». Se ha notado que la lógica difusa es más aplicable cuando el rol del usuario final se encuentra en un nivel gerencial.*

B. Diseño

En el diseño, además de las actividades propias de la metodología, es necesario especificar y analizar las consultas difusas a realizar. Las especificaciones se hacen en lenguaje natural pues es allí donde se usan términos vagos. Una vez especificadas las consultas, se estudian los elementos de naturaleza vaga y se modelan mediante la teoría de conjuntos difusos. En esta modelación se determina cuáles serán las variables lingüísticas, así como, los términos difusos que se usarán, determinando su tipo y forma. Se dan también valores por defecto para estos términos. Recuerde que la interpretación de cada término podría variar de acuerdo a las preferencias del usuario. En la mayoría de los casos, una palabra de naturaleza vaga del lenguaje natural se modela como un solo término difuso, sin embargo, en algunos casos es necesario usar una expresión más compleja que involucra varios términos difusos.

Los elementos lingüísticos vagos a considerar en el diseño, y su correspondiente modelación, son los siguientes: ¾ Adjetivos calificativos de grado positivo, éstos añaden una nota de cualidad. El grado positivo corresponde al adjetivo en su forma original. Ejemplos de éstos son *bueno, malo, grande, pequeño, alto, bajo, externo, interno, joven, viejo, largo, corto, barato, caro, eficaz, productivo, nuevo, antiguo, pobre, rico, simple, complejo, pesado* y *liviano*. Generalmente son modelados como predicados difusos. ¾ Adjetivos calificativos de grado comparativo, éstos expresan superioridad o inferioridad, tales como *más joven, menos productivo, más eficaz, mejor, peor, mayor, menor, superior, inferior, exterior* e *interior*. Estos elementos lingüísticos se modelan como comparadores difusos. ¾ Adjetivos calificativos de grado superlativo relativo, éstos indican una superioridad o inferioridad relativa a un grupo de entes similares. Se forman con los adjetivos comparativos añadiendo un artículo determinado, así como *el más productivo, el menos*. También hay algunos puros como: *óptimo, pésimo, máximo, mínimo, supremo, sumo, ínfimo, extremo* e *íntimo*. La modelación de estos términos requiere de un cuantificador y un comparador difuso. ¾ Adjetivos calificativos de grado superlativo absoluto, estos denotan la superioridad o inferioridad ideal, se caracterizan pos las terminaciones *ísimo* y *érrimo*, tales como *buenísimo, malísimo, novísimo, viejísimo, antiquísimo, grandísimo, pequeñísimo, altísimo, bajísimo, riquísimo,* y *paupérrimo*. Éstos tienen equivalencia con el uso del adverbio *muy* seguido de un adjetivo calificativo positivo. Puede modelarse como un predicado difuso posiblemente afectado por un modificador difuso. ¾ Adjetivos determinativos indefinidos cuantitativos, éstos añade una nota vaga sobre la cantidad, tales como *algunos, bastantes, cuantiosos, demasiados, muchos, múltiples, ningunos, numerosos, pocos, tantos, todos, unos* y *varios*. Naturalmente, éstos se modelan como cuantificadores difusos. ¾ Adverbios

variables, son palabras que modifican al verbo a un adjetivo u otro adverbio, particularmente interesan adverbios como *bien, mal, grandemente, pequeñamente, altamente, bajamente, externamente internamente, óptimamente, pésimamente, máximamente, mínimamente, supremamente, sumamente, ínfimamente* y *extremamente*, éstos se modelan como modificadores difusos. Otros ejemplos de este grupo son las palabras *cerca* y *lejos*, éstas son claramente representantes los comparadores difusos. ¾ Adverbios invariables de cantidad, son adverbios que no tienen grado, interesan palabras como *bastante, harto, medio, muy, poco, mucho, tanto, tan, casi* y *algo*. Éstos se modelan como modificadores difusos.

C. Implementación

Para la implementación del sistema se cuenta con SQLf. Éste provee sentencias para la definición de términos difusos. También tiene todas las estructuras de consulta y programación definidas por el estándar SQL con el añadido de poder usar una condición difusa en cualquier lugar donde SQL permite una condición Booleana. Estas estructuras se usarían para programar los requerimientos difusos. Existe un servidor de consultas difusas a bases de datos relacionales llamado SQLfi[7], es decir Servidor de SQLf para aplicaciones en Internet. SQLfi provee de una interfaz para programación de Aplicaciones (API) a través del protocolo de invocación de objetos remotos (RMI). SQLfi fue desarrollado en JAVA sobre ORACLE, sin embargo puede ser migrado a otros manejadores de bases de datos relacionales y puede ser usado desde aplicaciones en otros lenguajes de programación. Como respuesta a una consulta difusa, SQLfi retorna un conjunto resultado difuso que se implementa como un objeto serializable, éste no es más que la extensión difusa de los ResultSets del estándar de conexión a bases de datos de JAVA (JDBC). Esta extensión consiste en un campo extra por cada fila en el cual SQLfi deja el grado de satisfacción de la consulta difusa para esa fila.

V. CASO DE ESTUDIO

A fin de ejemplificar esta propuesta metodológica, se planteó el desarrollo de una aplicación Web para el manejo de consultas difusas sobre la Encuesta de Opinión Estudiantil (EOE) de la Universidad Simón Bolívar (USB), la cual fuese de apoyo en la toma de decisiones académicas a los diferentes miembros de la comunidad académica. Aquí se mostrarán sólo los aspectos relevantes del desarrollo de la aplicación que involucran el uso de consultas difusas de acuerdo con los aspectos de la propuesta metodológica. La USB tiene un sistema de EOE que registra la apreciación del estudiantado sobre los cursos y los docentes que los imparten en los diferentes períodos académicos. Cada período consta de un trimestre, dictándose tres trimestres en el año. Durante casi 10 años la base de datos de la encuesta de opinión estudiantil está poblada con alrededor de 4.000.000 de registros de opiniones. Toda esa información podría ser de gran utilidad para alumnos, docentes y gerentes universitarios en sus procesos de toma de decisión y en la mejora de la calidad de la enseñanza. Sin embargo tal potencialidad no ha sido explotada debido a que no se usan medios cercanos al lenguaje natural como lo serían las consultas difusas. Por ejemplo, un profesor podría querer saber cuáles son sus aspectos débiles como docente. Un estudiante podría necesitar conocer cuáles son las materias electivas que tienen mejores dotaciones de material de apoyo y mayores expectativas de éxito. Un gerente podría preguntarse cuáles son los profesores más destacados para darles un premio de estímulo. Estos requerimientos involucran términos agos que expresan preferencias de los usuarios. El instrumento de la EOE consta de 31 ítems cuyas respuestas están en la escala del 1 al 5. El estudiante puede dejar ítems sin contestar. Los primeros 19 ítems recogen la «Opinión sobre la actuación del Docente». Los ítems 20 y 21 son sobre la «Apreciación General» que el estudiante tiene del docente. Los ítems del 22 al 26 componen una sección de «Autoevaluación del estudiante». Los ítems del 27 al 31 permiten al estudiante expresar su «Opinión sobre el curso». Se escogió este caso debido a que es una necesidad real de darle mayor y mejor uso a estas encuestas, además resulta sencillo e ilustrativo el hecho de tener que las opiniones son expresadas como número enteros en el rango del 1 al 5 lo cual facilitará presentar modelos de términos difusos.

A. Análisis

Los actores que se detectaron para el sistema de consultas a la EOE fueron: el *Estudiante*, el *Docente*, el *Coordinador de Carrera* y el *Jefe de Departamento*. La identidad del usuario y su rol definen privilegios de acceso a los datos según restricciones impuestas a priori por la Dirección de Ingeniería de la Información de la USB (DII).

Existe una jerarquía entre los distintos actores. A medida que crece el nivel, se añaden nuevos casos de uso y se heredan los anteriores. Los casos de uso de la Figura. 4 son descritos en la Tabla I. El levantamiento de requerimientos se hizo en base a entrevistas con el Coordinador de la EOE, un profesor y dos estudiantes como usuarios. Se hizo una validación mediante encuestas abiertas a la comunidad académica. Analizando los casos de uso y los requerimientos no funcionales de la aplicación, en la Tabla II se evidencia que esta aplicación presenta las siete características que determinan la necesidad de usar lógica difusa.

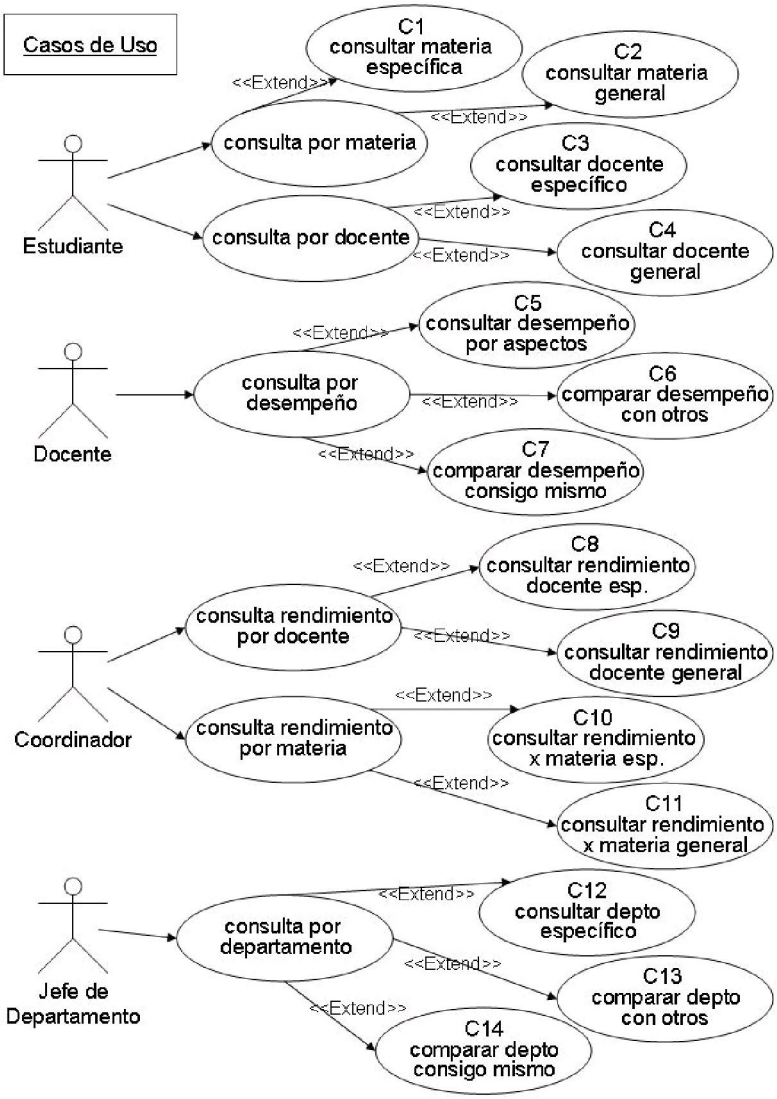

Figura 4. Diagrama de Casos de Uso

Tabla I. Descripción de los casos de uso

CU	DESCRIPCIÓN
C1	Consultar materia específica. El usuario escoge un departamento y una materia dentro de éste. Pregunta sobre nivel de dificultad de la materia, oportunidad de aprobación, disponibilidad de recursos, correspondencia con el número de créditos, calificación esperada.
C2	Consultar materia general. Mostrar lista de materias por período ordenadas según el nivel de satisfacción de criterios como los de C1.
C3	Consultar docente específico. El usuario escoge un departamento y un profesor dentro de éste. Se obtiene información en cuanto a su desempeño global del profesor y la manera como imparte sus materias.
C4	Consultar docente general. El usuario escoge un departamento. Mostrar lista de docentes ordenados de acuerdo al nivel de satisfacción de criterios como los de C3.
C5	Consultar desempeño por aspectos. Seleccionado un período, aquí se reflejarán los aspectos en los que el docente se destacó en determinado curso y aquellos que puede mejorar según la opinión de sus estudiantes.
C6	Comparar desempeño con otros. Seleccionado un período, el docente podrá comparar su desempeño con el de sus colegas, en las materias que imparte, de acuerdo a los aspectos de C5.
C7	Comparar desempeño consigo mismo. Seleccionado un período, el profesor podrá comparar su desempeño en con los períodos anteriores donde ha impartido uno o varios cursos, de acuerdo a los aspectos de C5.

C8	Consultar rendimiento de docente específico. El usuario tiene asociado según su perfil una carrera o departamento, dentro de las materias de esa unidad, se podrá observar el rendimiento de un docente dado, de acuerdo a los aspectos de C5.
C9	Consultar rendimiento de docente general. El usuario tiene asociado según su perfil una carrera o departamento. Mostrar un agregado de todos los profesores que imparten las materias de esa unidad o la lista de los profesores ordenada por el grado de satisfacción de un criterio, de acuerdo a los aspectos de C5.
C10	Consultar rendimiento por materia específica. El usuario tiene asociado según su perfil una carrera o departamento, y selecciona una materia que le pertenece para ver el rendimiento en cuanto a la preparación previa, la calificación esperada, el aprendizaje efectivo de los estudiantes, la dotación de materiales y equipos, así como, el esfuerzo requerido respecto al número de créditos de la misma.
C11	Consultar rendimiento por materia general. El usuario tiene asociado según su perfil una carrera o departamento. Mostrar un global por carrera o departamento o una lista ordenada de materias por grado de satisfacción de los criterios de C10.
C12	Consultar departamento específico. El usuario selecciona un departamento y un período. Se muestre el desempeño de un departamento en ese período de acuerdo a criterios que establezca el usuario.
C13	Comparar el departamento con otros. El usuario selecciona un departamento y un período. Se compara el desempeño de un departamento con respecto a los otros para ese período.
C14	Comparar el departamento consigo mismo. El usuario selecciona un departamento y un período. Se compara el desempeño del departamento con los períodos anteriores al seleccionado.

Tabla II. Cumplimiento de Características

Intuitividad	Los casos de uso de consultas de la aplicación van enfocados hacia responder presuntas intuitivas como ¿Cuáles son los profesores con *mejor desempeño*? ¿Cuáles son los cursos que tienen *mayor dificultad*? ¿Cuáles son los *mejores* departamentos de la USB, en cuanto al *nivel de rendimiento* de sus cursos impartidos? En todas estas preguntas podemos observar en *itálicas* los elementos lingüísticos importantes para la aplicación que el usuario los maneja en forma intuitiva.
Flexibilidad	Es importante considerar que el uso de la aplicación de consultas será para el apoyo a la toma de decisiones. Por ejemplo un estudiante al momento de inscribir una electiva, un profesor que desee cambiar el material de apoyo de un curso, un jefe de departamento que quiera proponer a un docente para un curso de mejoramiento. En todos estos casos se debe tomar una decisión con la información disponible, aunque no haya una respuesta que se ajuste completamente al ideal. Vale la pena entonces que el sistema sea flexible para incluir respuestas en el borde.
Vaguedad	En las preguntas que observamos cuando analizamos la intuitividad, vimos términos difusos como *mejor*, *bueno*, *muy*, *más*, *buen*; los cuales a su vez impulsan la necesidad de que también existan otros como *peor*, *malo*, *menos*, *fácil*, *difícil*, *promedio*, *excelente*, *regular*, *deficiente*. La presencia de estos términos muestra la relevancia de la vaguedad en la aplicación.
Tolerabilidad	Debido a la cantidad de información que almacena la EOE, es completamente pertinente el uso de grados de satisfacción en las consultas difusas, con el fin de no sólo garantizar la calidad de las respuestas sino también acotar la cantidad de resultados aceptados. El usuario podría especificar con una interfaz amigable un nivel de tolerancia o un número máximo de respuestas deseadas.
Adaptabilidad	Se observa que hay elementos lingüísticos vagos en la aplicación. En el diseño es posible dar un modelo general de interpretación de estos términos. Hay que considerar que el mismo término puede tener una acepción diferente en cada contexto. Por otro lado, es conveniente que la implementación prevea que los distintos usuarios podrían tener interpretaciones diferentes según sus preferencias. La aplicación debe proveer mecanismos que permitan esta variabilidad.
Gradualidad	El propósito de la aplicación es ayudar al usuario en su toma de decisiones que puedan ser afectadas por el desempeño docente. Para facilitar esto es conveniente que las respuestas sean dotadas de un grado de satisfacción de acuerdo con las preferencias del usuario.
Gerencialidad	En este caso se observa que el Jefe de Departamento y el Coordinador de Carrera, quienes tienen posiciones gerenciales en la organización son los que pueden actuar en mayor cantidad de casos de uso y que éstos son a la vez más complejos. Ellos pueden también tener visión global sobre el desempeño de profesores y/o materias a nivel de unidad y hacer comparaciones entre unidades.

B. Diseño

El diseño de la base de datos de la EOE fue realizado por la DII, fue un trabajo previo e independiente al desarrollo aquí presentado. La Figura. 5 contiene una simplificación del diagrama relacional de la base de datos de la EOE. Se han omitido algunas tablas y atributos que no son relevantes para las consultas difusas que se quiere ilustrar en este trabajo.

En la tabla «unidad_asignatura» hay una clave foránea «código» hacia su homónimo en «asignatura» la cual debido a que la DII diseñó esta referencia para ser usada con el operador LIKE de SQL y no con una igualdad estricta, de manera que un registro en «unidad_asignatura» puede referenciar a varios de «asignatura». Esa decisión de diseño incide en que se dificulte garantizar la integridad referencial y que las consultas usen un

operador LIKE en lugar de la igualdad (NATURAL JOIN). La razón de esta decisión obedeció al deseo de disminuir el número de filas de la tabla «unidad_asignatura» aprovechando el hecho que para el caso de los departamentos académicos, el nombre del departamento implica funcionalmente un prefijo en el código de la materia.

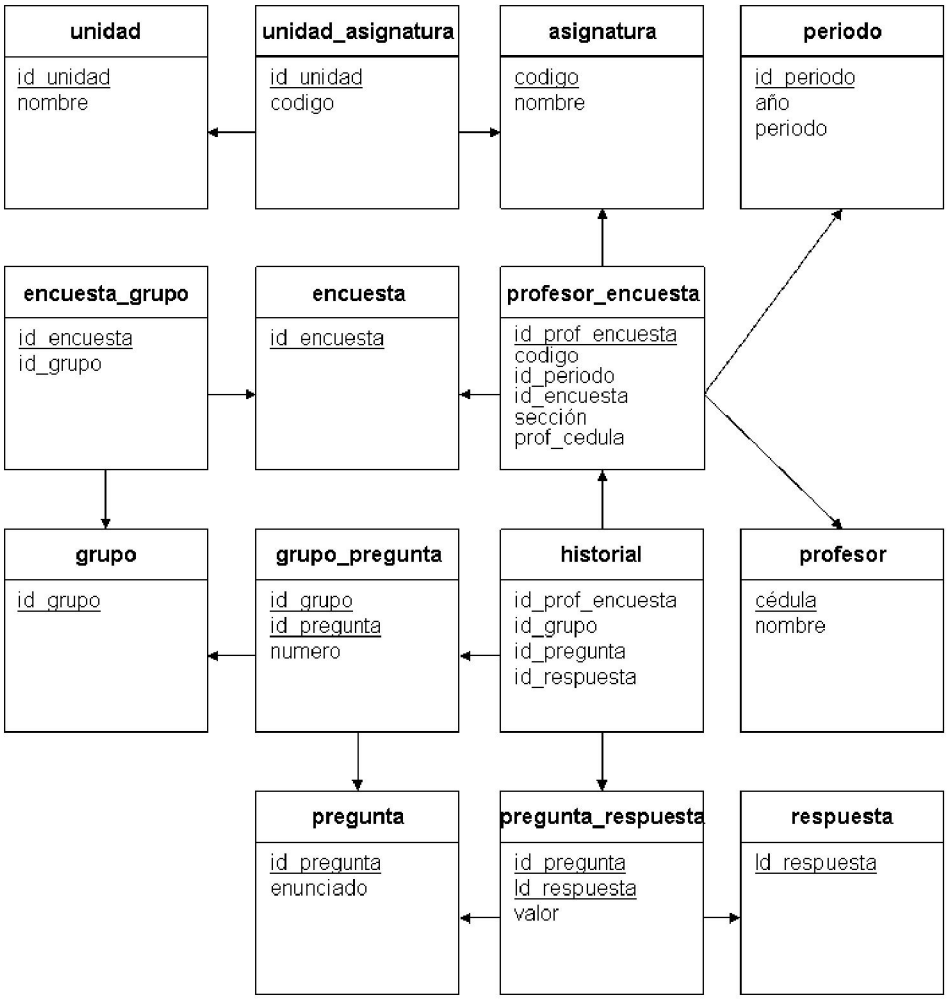

Figura 5. Base de Datos de la EOE, Diagrama Relacional

A efectos de la extensión metodológica que se propone en este artículo, la primera actividad que se añade al diseño es especificar las consultas en lenguaje natural, tal como está en la Tabla III. Debido a que el número de materias es muy alto para su manipulación, así como el número de profesores; todas las consultas asumen que el usuario ha seleccionado primero el departamento. Si el usuario es un profesor o un jefe de departamento, la selección corresponde a su departamento de adscripción. En las preguntas que involucran un profesor o una materia específica, el usuario lo seleccionaría de una lista restringida al departamento. Si el usuario es un profesor, entonces ya estaría él mismo como profesor seleccionado. En las consultas que involucran una comparación se asume que el usuario ha seleccionado previamente un período académico en el cual él quiere hacer la comparación. Estas selecciones serían filtros que se programan en la aplicación a través de consultas clásicas a la base de datos.

Una vez especificadas las consultas, se procede a determinar cuáles son los términos vagos y proponer su modelación en lógica difusa.

En la Tabla III se puede observar que las consultas tienen listas de selección de términos que se colocaron entre paréntesis. Haciendo la salvedad de la lista (departamento, materia) en la consulta C5.B, todos ellos son elementos lingüísticos vagos, por lo cual se han colocado en itálicas y negrillas para enfatizarlos.

Tabla III. Especificación de Consultas en Lenguaje Natural

C1.A	¿Cuáles de los profesores que imparten la materia M lo hacen de forma tal que ésta resulta (fácil, regular, difícil))?
C1.B	La materia M: ¿brinda una oportunidad de aprobación (***baja, regular, alta***)? ¿se puede esperar una calificación (***baja, regular, alta***)? ¿se puede considerar (***fácil, regular, difícil***)?
C2.A	¿Cuán (***fácil, regular, difícil***) resultan las materias?
C2.B	¿Cuáles son las materias donde ***la mayoría*** de los estudiantes espera obtener una calificación (***baja, regular, alta***)?
C2.C	¿En qué materias ***la mayoría*** de los estudiantes opinan que la oportunidad de aprobación es (***baja, regular, alta***)?
C3.A	¿Cuáles materias el profesor P imparte de forma tal que resultan (***fácil, regular, difícil***)?
C3.B	¿En qué medida el profesor P tiene un desempeño global (***bajo, regular, alto***)?
C4.A	¿Cuáles profesores imparten sus materias de forma tal que resultan (***fácil, regular, difícil***)?
C4.B	¿Cuáles profesores tienen un desempeño global (***bajo, regular, alto***)?
C5.A	¿Cuáles son los aspectos (***débiles, destacados***) del profesor P?
C5.B	¿En qué aspectos el profesor P es (***más débil, más destacado***) que los colegas del mismo (departamento, materia)?
C6.A	¿En cuáles materias el profesor P es (***más débil, más destacado***) respecto a los demás colegas?
C7.A	¿En cuáles aspectos el profesor P es (***más débil, más destacado***) respecto a periodos anteriores?
C7.B	¿En cuáles materias el profesor P es (***más débil, más destacado***) respecto a periodos anteriores?
C8,A	Para la materia M, con el profesor P ¿en qué medida la calificación esperada suele ser (***baja, regular, alta***)?
C8.B	¿En qué medida la calificación esperada con el profesor P suele ser (***baja, regular, alta***)?
C9.A	Para cada profesor ¿en qué medida la calificación esperada suele ser (***baja, regular, alta***)?
C9.B	En la materia M, para cada profesor ¿en qué medida la calificación esperada suele ser (***baja, regular, alta***)?
C10.A	¿En qué medida la materia M, cumple con el criterio C de la encuesta de forma (***baja, regular, alta***)?
C11.B	¿En cuáles materias se cumple el criterio C de forma (***baja, regular, alta***)?
C12.A	En el departamento D, ¿en qué medida la calificación esperada suele ser (***baja, regular, alta***)?
C13.A	La calificación esperada en el departamento ¿es (***peor, mejor***) que en los otros departamentos de la universidad?
C14.A	La calificación esperada en materias del departamento ¿es (***peor, mejor***) que en períodos anteriores?
C12.B	¿En que medida los profesores del departamento tienen un desempeño global (***bajo, regular, alto***)?
C13.B	En el departamento D ¿el desempeño global de los profesores es (***peor, mejor)*** que en los otros departamentos?
C14.B	En el departamento D ¿el desempeño global de los profesores es (***peor, mejor)*** que en los períodos anteriores?

Los términos ***bajo***, ***regular*** y ***alto*** (o ***baja***, ***regular*** y ***alta***) son sin duda los más sencillos. Éstos son adjetivos calificativos de grado positivo, por lo que su modelación perfectamente se haría como predicados difusos. Siendo que el universo es conjunto finito y manejable de elementos, es posible usar una definición por extensión. También, por ser un rango numérico, se pueden usar representaciones trapezoidales como las sugeridas en la Figura. 6. Se adoptarán éstas como las definiciones por defecto para estos predicados.

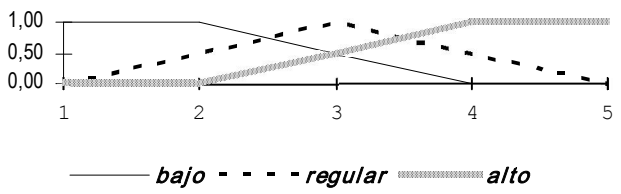

Figura 6. Modelo de los predicados *bajo, regular* y *alto*

El usuario podría personalizar los valores que definen estos predicados difusos, según su preferencia, para ello debe proveerse una interfaz de usuario final que lo facilite. La Figura. 7 muestra un diseño de interfaz para los predicados difusos ***bajo***, ***regular*** y ***alto***. En esta interfaz se ha utilizado como metáfora una regleta en la cual los deslizadores del borde inferior señalan el intervalo abierto que define el soporte del conjunto difuso mientras que los marcadores deslizables del borde superior señalan el intervalo cerrado que define el núcleo. El hecho que en un caso el intervalo es abierto y en el otro es cerrado se representa al usar símbolos diferentes, los del borde inferior están sobre una pequeña caja como queriendo señalar que «no toca el borde», es decir el intervalo es abierto. La selección mostrada en interfaz de la Figura. 7 da exactamente la definición de la Figura. 6. Interfaces similares se pueden usar para cualquier otro término que sea modelado mediante un conjunto difusos de función de membresía trapezoidal.

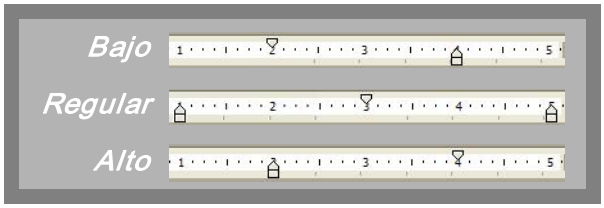

Figura 7. Interfaz para especificación los predicados *bajo, regular* y *alto*

Los términos ***peor*** y ***mejor*** son adjetivos calificativos de grado comparativo, lo que indica que la presencia de comparadores difusos. En este caso si se decide modelarlos como comparadores definidos por extensión, se tendría que listar a lo sumo 25 pares ordenados con sus respectivos grados, dado que el universo es el rango entero del 1 al 5. A pesar de que dicha lista no es tan larga, por tratarse de un dominio numérico, es más ilustrativa y compacta la definición mediante una distancia. En este caso, siendo que los elementos del universo están en un mismo orden de magnitud, resulta conveniente adoptar la diferencia entre los elementos comparados. Si esta distancia es 0 se puede decir que ninguno es *peor que* (o *mejor que*) el otro, y a medida que esta distancia crece el primero es *peor que* el segundo (o el segundo es *mejor que* el primero). La función de membresía puede ser de forma trapezoidal como en la Figura. 8.

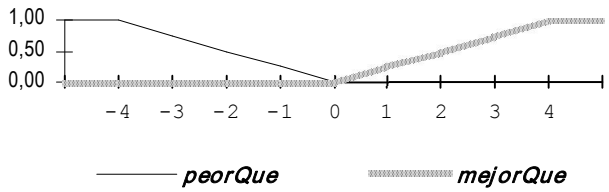

Figura 8. Definición trapezoidal de los comparadores *peorQue, mejorQue*

Los términos ***más débil*** y ***más destacado*** son también adjetivos calificativos de grado comparativo y se modelarán con los mismos comparadores que los términos ***peor*** y ***mejor***.

Los términos ***débil***, y ***destacado*** son adjetivos calificativos de grado positivo. Sin embargo, en el contexto se podrían considerar como equivalentes a los términos ***bajísimo*** y ***altísimo***, los cuales son adjetivos calificativos de grado superlativo absoluto. No se usarán estos términos en la expresión de las consultas porque su uso no es políticamente aceptable para describir aspectos del desempeño de un profesor. Se modelará ***destacado*** mediante la expresión lingüística ***bastante alto.*** Aquí ***bastante*** es un modificador difuso definido como una traslación de longitud 1 en el eje de las abscisas en sentido negativo. Si el predicado ***alto*** tiene la definición de la Figura. 6, el modificador ***bastante*** tendría el efecto mostrado en la Figura. 9. El predicado ***débil*** se modelará mediante la negación del predicado ***destacado***, es decir ***no destacado***. Como la negación es interpretada mediante el complemento a uno, se obtienen la función de membresía que se muestra en la Figura. 9.

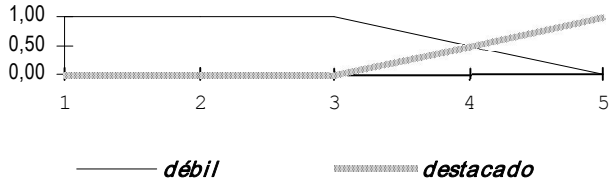

Figura 9. Modelo de los predicados *destacado (bastante alto) y débil (no destacado)*

Los términos ***fácil***, ***regular***, ***difícil*** son adjetivos calificativos de grado positivo. Al analizar la encuesta de opinión estudiantil se observa que no hay un ítem específico que pueda ser usado como variable para determinar en qué grado la materia cumple este criterio. Se pensó en definir predicados compuestos para esto. Por ejemplo, una materia es ***fácil*** cuando cumple con tres criterios: la preparación previa que tienen los estudiantes es ***alta***, la dedicación requerida para aprobarla es ***baja*** y la disponibilidad de materiales de apoyo es ***alta***. De forma similar se definen los términos **regular** y **difícil** para el contexto materia (Tabla IV).

Tabla IV. Definiciones de los términos DIFUSOS fácil, regular, difícil

TÉRMINO	DEFINICIÓN (CONDICIÓN DIFUSA)
fácil	Preparación Previa = alta ∧ Dedicación Requerida = baja ∧ Disponibilidad de Materiales = alta
regular	Preparación Previa = regular ∧ Dedicación Requerida = regular ∧ Disponibilidad de Materiales = regular
difícil	Preparación Previa = baja ∧ Dedicación Requerida = alta ∧ Disponibilidad de Materiales = baja

El término ***la mayoría*** aparece en forma explícita en dos de las consultas especificadas. Sin embargo, dado que todas las consultas están sintetizando las opiniones de todos los estudiantes que han contestado la encuesta, se podría decir que este término está implícitamente en la naturaleza de todas estas consultas. Por ejemplo, la pregunta *¿Cuáles materias el profesor P imparte de forma tal que resulta* ***(fácil, regular, difícil)****?* se puede parafrasear como *¿Cuáles materias el profesor P imparte de forma tal que resultan* ***(fácil, regular, difícil)****, según la opinión de la mayoría de los encuestados?*. La expresión lingüística ***la mayoría*** es un adjetivo determinativo indefinido cuantitativo, por lo que corresponde a un cuantificador difuso, además, es proporcional pues describe una cantidad relativa a un todo. Su modelación podría ser como aquélla de la Figura. 3 en la sección titulada «Fundamentos de SQLf»

C. Implementación

Una vez determinadas las consultas y modelados los términos difusos que ellas involucran, se procede a implementarlas

usando SQLf. Es pertinente recordar que éste es un lenguaje de consulta a bases de datos y no un lenguaje de propósito general. SQLf puede ser utilizado por otro leguaje anfitrión (como JAVA) a través de una interfaz de programación de aplicaciones provista por el manejador de consultas difusas. Aquí se mostrará sólo la implementación de las consultas en SQLf y no de la aplicación completa.

En este caso se requiere definir en SQLf tres predicados difusos, dos comparadores difusos y un cuantificador difuso, lo cual puede hacerse así:

```
CREATE FUZZY PREDICATE bajo ON 1..5
AS (INFINITE, INFINITE, 2, 4)
CREATE FUZZY PREDICATE regular ON 1..5
AS (1, 3, 3, 5)
CREATE FUZZY PREDICATE alto ON 1..5
AS (2, 4, INFINITE, INFINITE)
CREATE FUZZY MODIFIER bastante
AS TRANSLATION -1
CREATE FUZZY COMPARATOR peorQue ON 1..5
AS (x-y) IN (INFINITE, INFINITE, -4, 0)
CREATE FUZZY COMPARATOR mejorQue ON 1..5
AS (x-y) IN (0, 4, INFINITE, INFINITE)
CREATE FUZZY QUANTIFIER laMayoría
AS (0.3333, 0.8333, 1.0, INFINITE)
```

Los términos ***débil***, ***destacado***, ***fácil***, ***regular*** y ***difícil*** no requieren definiciones pues ellos se definen como expresiones complejas que utilizan algunos de los términos ya definidos. Su especificación es directa en la consulta SQLf de acuerdo con la modelación hecha.

En la Figura. 5 se muestra el diseño base de datos de la EOE. Puede observarse que la información principal de las opiniones de los encuestados, requerida para resolver las consultas se encuentra dispersa en varias tablas. Es por esto que resulta conveniente la creación de una vista la cual se denominará «principal».

```
CREATE VIEW principal AS
SELECT
pe.id_prof_encuesta, pe.codigo, pe.id_periodo, pe.id_encuesta, pe.seccion, pe.prof_cedula, h.id_grupo, h.id_pregunta, h.id_respuesta,
pr.valor, pr.descripcion, gp.numero
FROM
historial AS h, profesor_encuesta AS pe,
pregunta_respuesta AS pr, grupo_pregunta AS gp
WHERE h.id_pregunta=pr.id_pregunta
AND h.id_respuesta=pr.id_respuesta
AND pe.id_prof_encuesta=h.id_prof_encuesta
AND gp.id_pregunta=pr.id_pregunta
```

Los atributos «numero» y «valor» de la vista principal son esenciales para las consultas pues ellos contienen respectivamente el número que identifica el ítem de la encuesta y el valor de la respuesta dada por el estudiante que emitió su opinión. Nótese que para un mismo formulario de encuesta aplicado a un estudiante para una materia y período específico, los distintos ítems aparecen en diferentes filas de la vista principal.

A continuación se muestra la implementación en el lenguaje SQLf de algunas de las consultas del caso de estudio. Se asume que el usuario ha seleccionado un departamento, el cual se encuentra en la variable $sel1. También, en las consultas que sea necesario, el usuario habrá seleccionado un profesor y un período, variables $sel2 y $sel3, respectivamente. Se asume que el nivel de calibración cualitativa, especificado por el usuario, está en la variable $sel4.

La consulta C2.B de la Tabla III «¿Cuáles son las materias donde ***la mayoría*** de los estudiantes espera obtener una calificación ***alta***?» se implementa mediante la consulta en SQLf:

```
SELECT codigo FROM
unidad_asignatura AS ua,
principal AS p
WHERE ua.id_unidad = $sel1
AND p.codigo IS LIKE ua.codigo
AND p.numero = 25
GROUP BY p.codigo
HAVING laMayoria ARE p.valor = alto
WITH CALIBRATION $sel4
```

En este ejemplo, aparece la variable lingüística *Calificación Esperada*, que corresponde al ítem número 25 de la encuesta. La cláusula GROUP BY está particionando la encuesta por materias. El cuantificador difuso *laMayoría* indica la proporción de filas en la clase de partición que se espera que cumplan la condición difusa *p.valor = alto*. En otras palabras, que la mayoría de las encuestas digan que la *Calificación Esperada* para esa materia es *alta*.

La consulta C5.A de la Tabla III «¿Cuáles son los aspectos ***débiles*** del profesor?», siendo que en las encuestas estos aspectos corresponden a los ítems 01 al 19, se implementaría en SQLf así:

```
SELECT p.variable FROM principal AS p
WHERE p.prof_cedula = $sel2
AND p.id_periodo = $sel3
AND p.numero BETWEEN 01 AND 19
GROUP BY p.numero
HAVING laMayoria ARE p.valor = not bastante alto
WITH CALIBRATION $sel4
```

La consulta C6.A de la Tabla III «¿En cuáles materias el profesor P es ***más destacado*** respecto a los demás colegas?», considerando el desempeño global que es el ítem 20 de la encuesta, se implementaría en SQLf así:

```
SELECT p1.codigo FROM
```

```
principal AS p1, principal AS p2
WHERE p1.prof_cedula = $sel2
AND p2.prof_cedula<>$sel2
AND p1.codigo = p2.codigo
AND p1.numero = 20
AND p2.numero = 20
GROUP BY p1.codigo
HAVING laMayoria ARE (p1.valor peorQue p2.valor)
WITH CALIBRATION $sel4
```

La consulta C2.A de la Tabla III «¿Cuán ***fácil*** resultan las materias?», se implementaría así:

```
SELECT codigo FROM
unidad_asignatura AS ua, principal AS p
WHERE ua.id_unidad = $sel1
AND p.codigo IS LIKE ua.codigo
AND numero IN {22,23,24}
GROUP BY codigo
HAVING laMayoria ARE
((numero IN {22,24} AND valor = alto)
OR (numero = 23 AND valor = bajo))
```

Recuerde que el predicado fácil se modeló en la Tabla IV como la combinación «Preparación Previa = ***alta*** Ù Dedicación Requerida = ***baja*** Ù Disponibilidad de Materiales = ***alta***». La Preparación Previa es el ítem número 22 de la encuesta, la Dedicación Requerida es el 23 y la Disponibilidad de Materiales el 24. Nótese que en la cláusula HAVING, bajo el alcance del cuantificador ***laMayoria*** las condiciones se vinculan con OR en lugar de AND. Esto se debe al diseño de la base de datos en el cual cada opinión de cada ítem se encuentra en una fila diferente de la tabla. Si se combina con AND el resultado sería vacío pues una misma fila no puede ser a la vez del criterio «Preparación Previa» y del criterio «Dedicación Requerida» y del criterio «Disponibilidad de Materiales», ya que ellos se encuentran en filas diferentes de la tabla, debido al diseño de la base de datos. La verificación de la condición usando OR puede hacerse con una prueba lógica mediante manipulación simbólica.

Una alternativa a esta expresión disyuntiva pudo haber sido la implementación de una conjunción vertical. Para ello se tendría que definir una nueva vista «principal_C2_A» para la combinación de las filas deseadas la cual podría ser así:

```
CREATE VIEW principal_C2_A AS
SELECT
p22.codigo,
p22.valor AS valor_p22,
p23.valor AS valor_p23,
p24.valor AS valor_p24
FROM
principal AS p22,
principal AS p23,
principal AS p24
WHERE p22.numero=22
AND p23.numero=23
AND p24.numero=24
AND p22.id_respuesta = p23.id_respuesta
AND p23.id_respuesta = p24.id_respuesta
```

El costo computacional del cálculo de esta vista podría ser elevado debido a que la vista «principal» contiene la tabla «historial» que recoge las opiniones emitidas por cada encuestado en cada planilla para cada pregunta durante todos los años de la encuesta (alrededor de cuatro millones de registros a la fecha actual).

De esta manera la expresión de la consulta sería la siguiente:

```
SELECT codigo FROM
unidad_asignatura AS ua, principal_C2_A AS p
WHERE ua.id_unidad = $sel1
AND p.codigo IS LIKE ua.codigo
GROUP BY codigo
HAVING laMayoria ARE
(valor_p22 = alto
AND valor_p23 = bajo
AND valor_p24 = alto)
```

VI. CONCLUSIONES

La principal contribución de este trabajo ha sido añadir a los modelos de desarrollo de software los elementos necesarios para el desarrollo de aplicaciones que requieran el uso de consultas difusas sobre una base de datos. Las extensiones propuestas se encuentran, según el modelo de ciclo de vida, en las fases de análisis, diseño e implementación. Se ha mostrado mediante una aplicación real el uso de estos aspectos metodológicos. Con este trabajo se abre el camino hacia el desarrollo de una nueva generación de sistemas de información más flexible, capaces de manejar preferencias de usuarios y dar resultados graduales. En trabajos futuros podría incorporarse esta propuesta a herramientas asistidas de ingeniería de software y ambientes de programación. Adicionalmente, sería conveniente la creación de un ambiente de programación tipo «*framework*» que facilite la implementación de aplicaciones que usen consultas difusas en SQLf.

AGRADECIMIENTOS

Este trabajo cuenta con el apoyo de FONACIT Venezuela Proyecto G-200500278 y de IRISA/ENSSAT Francia Proyecto Pilgrim. Agradecemos a Dios, nuestra inspiración y fortaleza, dejamos esta reflexión: «Y si no os parece bien servir al SEÑOR, escoged hoy a quién habéis de servir: si a los dioses que sirvieron vuestros padres, que estaban al otro lado del río, o a los dioses de los amorreos en cuya tierra habitáis; pero yo y mi casa, serviremos al SEÑOR.» (Josué 24:15, La Biblia de las Américas)

REFERENCIAS

[1] Beck K., 1999. Extreme Programming Explained: Embrace Change. Addison Wesley. Primera Edición.

[2] Bordogna G.; Psaila G, 2008. Customizable Flexible Querying Classic Relational Databases. En: Galindo J. (Ed.), (2008). Handbook of Research on Fuzzy Information Processing in Databases. Hershey, PA, USA: Information Science, Chapter VIII, pp 189-215.

[3] Bosc P; Pivert O., 1995. SQLf: A Relational Database Language for Fuzzy Querying. En: IEEE Transactions on Fuzzy Systems, Vol. 3, pp 1-17.

[4] Corcos D., 2000. El Modelo Espiral. Cuaderno de Reportes Técnicos en Ingeniería del Software, Vol. 2, No. 3. Editorial CAPIS.

[5] Galindo J.; Urrutia A.; Piattini M., 2006. Fuzzy Database Modeling, Design and Implementation, Idea Group Publishing.

[6] Galindo J., 2008. Introduction and Trends to Fuzzy Logic and Fuzzy Databases. En: Galindo J. (Ed.), (2008). Handbook of Research on Fuzzy Information Processing in Databases. Hershey, PA, USA: Information Science, Chapter I, pp 1-33.

[7] Goncalves M.; Tineo L., 2008. SQLfi y Sus Aplicaciones. En: Revista Avances en Sistemas e Informática, Vol. 5, No. 2.

[8] González C., 2008. Una propuesta para la integración y estandarización de SQLf y FSQL. Trabajo Especial de Grado de Maestría, Universidad Simón Bolívar, Caracas, Venezuela.

[9] Kruchten P., 2003. The Rational Unified Process: An Introduction. 3rd Edition. Addison-Wesley.

[10] NASA, 1998. Formal Methods Specification and Verification Guidebook for Software and Computer Systems. Volume I: Planning and Technology Insertion. National Aeronautics and Space Administration, USA.

[11] Rodríguez R.; Tineo L., 2009. Elementos Gramaticales y Características que Determinan Aplicaciones con Requerimientos Difusos. En: Revista Teckhne. Universidad Católica Andrés Bello. Caracas, Venezuela, Vol. 12, No. 1.

[12] Sommerville I., 2007. Ingeniería de Software. 8va. edición. Addison Wesley.

[13] Tineo L, 1998. Interrogaciones Flexibles en Bases de Datos Relacionales. Trabajo de Ascenso, Universidad Simón Bolívar, Caracas, Venezuela.

[14] Tineo L., 2003. A fuzzy quantifiers' interpretation for database querying. En: Proceedings of FIP 2003.

[15] Tineo L., 2006. Una contribución a la interrogación flexible de bases de datos: Evaluación de Consultas Cuantificadas Difusas. Tesis Doctoral, Universidad Simón Bolívar, Caracas, Venezuela.

[16] Vaughn R. Application of Lightweight Formal Methods in Requirement Engineering, Mississippi State University, Mississippi, USA.

[17] Wikipedia, 2009. La enciclopedia libre: Modelo en Cascada. http://es.wikipedia.org/wiki/Modelo_en_cascada

[18] Yager R., 1991. Connectives and quantifiers in fuzzy sets. En: Fuzzy Sets and System, Vol. 40, pp. 39-76.

[19] Zadeh L. A., 1975. The concept of a linguistic variable and its application to approximate reasoning. En: Information Science, Vol. 8, pp 199-249.

[20] Zadeh L. A., 1965. Fuzzy Sets. En: Information and Control, Vol. 8, No. 3, pp. 338-353.

Marlene Goncalves PhD. en Computación, Universidad Simón Bolívar, Caracas, Venezuela, (USB) 2009, MSc. en Ciencias de la Computación, USB 2002. Lic. en Computación, Universidad Central de Venezuela, Caracas, Venezuela, 1998.

Es Profesor Asociado (desde 2007), miembro del Personal Académico de la Universidad Simón Bolívar (desde 2001). Tiene la acreditación del Programa de Promoción al Investigador del OCT de Venezuela como Investigador Nivel 1 (desde 2007). Ha recibido la distinción Profesor Meritorio de la CONABA (2002), Es Jefe del Laboratorio de Bases de Datos de la Universidad Simón Bolívar (desde 2006). En el ámbito de la preferencia de Gestión de Bases de Datos, tiene más de diez artículos en extenso, más de cinco publicaciones de notas breves, tres documentos en revistas indexadas y más de diez direcciones de trabajos conducentes a títulos académicos. Es co-responsable del proyecto «Creación y Aplicación de Manejadores de Bases de Datos Difusas» con el apoyo de FONACIT (2006-2009).

Rosseline Rodríguez. MSc en Ciencias de la Computación, Universidad Simón Bolívar (USB), Caracas, Venezuela, 1995. Ing. en Computación, USB 1991.

Es Profesor Agregado (desde 2003), Miembro del Personal Académico de la USB (desde 1991). Tuvo la Acreditación del Programa de Promoción al Investigador (PPI) del ONCTI como Investigador Candidato (1996). Ha recibido la distinción Profesor Meritorio de la CONABA (1998), En el área de Ingeniería de Software tiene alrededor de siete artículos in extenso en memorias arbitradas de congresos internacionales y alrededor de diez tutorías de trabajos conducentes a títulos académicos. Título de su Trabajo de Maestría «Un modelo funcional formal de sistemas y su equivalencia con Redes de Petri Coloreadas Extendidas», Título de su Proyecto de Grado «Sistema de Control de Consultorios Médicos». Miembro del personal de desarrollo del proyecto «Creación y Aplicación de Manejadores de Bases de Datos Difusas» auspiciado por FONACIT (2008-2009).

Leonid Tineo (Caracas Venezuela, 1968). PhD en Computación, Universidad Simón Bolívar (USB), Caracas, Venezuela, 2006. MsC en Ciencias de la Computación, USB, 1992. Ing. en Computación (Cum Laude), USB, 1990.

Es Profesor Titular (desde 2007), miembro del personal académico de la USB (desde 1991),Profesor Invitado en la ENSSAT Université de Rennes 1, Lannion, France (2009). Tiene una Acreditación del Programa de Promoción al Investigador de Venezuela como Investigador Nivel 1 (desde 2003). Ha recibido las distinciones: Profesor Destacado CONABA (2002), Destacada Labor Docente USB (1999). Fue Coordinador del Grupo de Investigación y Desarrollo en Bases de Datos de la USB (2002-2008). Ejerció el puesto de Coordinador de Información e Integración del Decanato de Investigación y Desarrollo en la USB (2002-2007). En el área de Bases de Datos Difusas, tiene alrededor de treinta contribuciones en Memorias de Congresos Arbitrados, más de quince notas breves publicadas, alrededor de veinte artículos en Revistas Indexadas, un capítulo de libro y más de quince tutorías de trabajos conducentes a títulos académicos. Fue responsable del Proyecto FONACIT G-2005000279 «Creación y Aplicación de Manejadores de Bases de Datos Difusas» (2006-2009). Es responsable del Grupo Venezolano en la Propuesta CYTED P509RT0108 Red Iberoamericana de Investigación en «Procesamiento Inteligente de Información y Conocimiento de Fuentes Heterogéneas en Ambientas con Web Semántica y Computación Ubicua» (2009).

Universidad Nacional de Colombia Sede Medellín
Facultad de Minas

Escuela de Ingeniería de Sistemas

Misión

La misión de la Escuela de Ingeniería de Sistemas es fomentar y apoyar la generación o la apropiación de conocimiento, la innovación y el desarrollo tecnológico en el área de ingeniería de sistemas e informática sobre una base científica, tecnológica, ética y humanística.

Visión

La formación integral de profesionales desde el punto de vista científico, tecnológico y social que les permita adoptar, aplicar e innovar conocimiento en el campo de los sistemas e informática en sus diferentes aspectos, aportando con su organización, estructuración, gestión, planeación, modelamiento, desarrollo, procesamiento, validación, transferencia y comunicación; para lograr un desempeño profesional, investigativo y académico que contribuya al desarrollo social, económico, científico y tecnológico del país.

Escuela de Ingeniería de Sistemas
Dirección Postal:
Carrera 80 No. 65 - 223 Bloque M8A
Facultad de Minas. Medellín - Colombia
Tel: (574) 4255350 Fax: (574) 4255365
Email: esistema@unalmed.edu.co
http://pisis.unalmed.edu.co/